テクニカル・リファレンス

(DOS/V)

バージョン6 対応

PC オープン・アーキテクチャー推進協議会

はじめに

パーソナル・コンピューターの普及が目覚ましく、その市場は依然として高成長が期待されます。一方、パーソナル・コンピューターのソフトウェアの共通利用に関しては、必ずしもユーザーの要求に十分応えていないというのが現状です。その根本的な理由は、ハードウェアの仕様がメーカー間で異なっているためですが、それに加えてパーソナル・コンピューターの場合、その開発経験や利用のノウハウの点で、あまりにも多岐にわたるからでもあります。

このような状況に鑑み、ソフトウェア利用の共通基盤の確立を主要な目的の一つとして『PCオープン・アーキテクチャー推進協議会（通称OADG)』が設立されました。これにより、異なるハードウェア上で稼働する多様なアプリケーションの提供が可能となり、この結果パーソナル・コンピューターの活用度が向上することになります。

OADGでは平成3年に「OADGテクニカル・リファレンス」として、関係の資料をハードウェア・テクニカル・リファレンスとソフトウェア・プログラミング・ガイドを内容とするものを発行いたしました。今回 第2回の改訂版として、OADGハードウェアおよびOADG DOS/V ソフトウェア関係の資料を、次の内容でより充実したものに改めました。

OADGテクニカル・リファレンス（ハードウェア）

「OADGハードウェア・インターフェース技術解説編」

OADGテクニカル・リファレンス（DOS/V)

「DOS/V 技術解説編」
「DOS/V BIOSインターフェース技術解説編」
「DOS/Vマウス・ドライバー技術解説編」
「DOS/Vプログラミング解説編」
「DOS/Vオプション機能 技術解説編」

OADG DOS/Vとは、このOADGテクニカル・リファレンスに基づいて作られたDOSで、OADG会員会社が採用および提供したものです。

OADGが目指すよりオープンな世界の確立のために、このテクニカル・リファレンスがお役に立てば幸いです。

平成6年12月

PCオープン・アーキテクチャー
推進協議会（OADG）

OADGテクニカル・リファレンス改訂作業参加者

亀井伸夫、坂本哲也、篠宮 誠 、渋谷尚亮、南部俊史、美根宏昭、若月文彦
（氏名五十音順）

用語の統一

OADGテクニカル・リファレンスでは次の用語に統一しています。

OADG DOS/V (OADGテクニカル・リファレンスに基づいて作られたDOSで、OADG会員会社が採用および提供したもの)
DOS/V (OADG DOS/Vの略称、すべてのバージョンを含む)
DOS/V 5.0（バージョン 5.0 を強調するときに)
DOS/V 6（バージョン 6 を強調するときに)

参考資料

OADGではプリンターの仕様をESC/P-J84と定めています。

EPSON ESC/P リファレンス・マニュアル 第2版、（セイコーエプソン（株））

次のような参考資料もご利用ください。

IBM Personal System/2 and Personal Computer BIOS Interface Technical Reference, (IBM S68X-2341)

Technical Reference Personal Computer AT, (IBM S229-9611, S229-9608)

Personal System/2 Hardware Interface Technical Reference — AT Bus System, (IBM S85F-1646)

一般図書

DOS/Vソフトウェアおよびハードウェア開発のサポートご案内、(OADG)

OADG CATALOG (OADG)

THE IBM PC & PS/2 プログラマーズガイド、（ピーター・ノートンとリチャード・ウィルトン著、翔泳社）

PC & PS/2 ビデオ・システム プログラマーズガイド、（リチャード・ウィルトン著、翔泳社）

THE IBM PC & PS/2 プログラマーズガイド、（ピーター・ノートンとリチャード・ウィルトン著、翔泳社）

プログラマーのためのPCソースブック、（トム・ホーガン著、翔泳社）

DOS/Vプログラミング・ガイド、（アスキー出版局）

MS-DOS 5 ブック、（アスキー出版局）

DOS/V magazine, （ソフトバンク　出版事業部、月刊）

この他にお気付きの一般図書がありましたら、OADG事務局までお知らせください。

テクニカル・リファレンス

DOS/V 技術解説編

目次

第2部　プログラミング・ユーティリティーの使用

第3部 付録

特記事項

「DOS/V 技術解説編」は、インターナショナル・ビジネス・マシーンズ・コーポレーション（以下IBMと略します）の著作物です。これはIBM社の出版物「IBM DOSバージョンJ5.0/V 技術解説書」および「PC DOS J6.1/V 技術解説書」の内容を記載しており、この著作権はIBMが所有しています。

これらの内容の無断転載、複製などの著作権に抵触する使用はできません。

本書で使用されている次の用語は、米国IBM社の商標です。

IBM
Personal Computer AT (略称：PC/AT)
Personal System/2 (略称：PS/2)
Personal System/55 (略称：PS/55)

第1章　入門

この章では本書について、および以下の項目に関しての情報がまとめられています。

- 本書の構成について

本書の構成

本書はDOS/V上でアプリケーション・プログラムを開発するのに必要な項目について説明します。

さらに、本書では読者がデバイス・ドライバーを作成したり、またシステムの拡張機能を使用してオペレーティング・システムを最適に利用する方法についても説明します。

各章は特定の題目について記述してあります。プログラムの作成または問題を解決する場合には、本書全体を読む必要はありません。また鍵となる項目は、目次を参照して見付け出すこともできます。

付録には参照情報が載せてあります。内容は、割り込み、ファンクション・コール(機能呼び出し)、およびデバイス・ドライバー・サービスを含むDOS/Vサービスの全リストです。

第1部　プログラムの作成

第2章　ディスクへのアクセス

この章では、読者が次タスクを完了するために必要なガイドおよびシステム・アーキテクチャーに関して説明します。

- ディスクのフォーマット
- ディスクへのアクセス
- ディスク間のデータの読み取りおよび書き込み

ディスクのフォーマット

DOS/Vでフォーマットされたすべてのディスクおよびディスケットはセクター・サイズが512バイトになります。DOS/Vはディスケット上で、ハード・ディスクの指定された区画を、次の順序でフォーマットします。

DOS/V構成要素	**サイズ**
ブート・レコード	１セクター
FAT (ファイル割り振りテーブル)の第１コピー	可変
FATの第２コピー	可変
ディスクのルート・ディレクトリー	可変
データ領域	可変

ブート・レコード

ブート・レコードはDOS/VのFORMATコマンドを使って作成します。ディスケットの場合、ブート・レコードはトラック０、セクター１、サイド０に常駐します。一方、ハードディスクに対しては、区画の開始セクターにブート・レコードが常駐します。有効なブート・レコードを持っていない媒体(ディスケットまたはハードディスク)にアクセスするとエラー・メッセージが発生します。

FAT (ファイル割り振りテーブル)

FAT (File Allocation Table: ファイル割り振りテーブル)は、ブート・レコードの直後に続いているセクターを占有します。FATのサイズが１セクター以上のときは、連続したセクター番号を占有します。

FATはディスク上の全ファイルの物理的な位置を管理しています。ディスクにエラーがあってFATの読み取りができないと、対象とするファイルの内容の位置を突き止めることができなくなります。このためディスクにはFATのコピーが２つ書き込まれます。

DOS/VはFATを使用して一回に１クラスターの割合でファイルにディスク空間を割り振ります。FATはディスクの各クラスターごとに12ビットのエントリー(1.5バイト)か、16ビットのエントリー(２バイト)で構成されています。ハード・ディスクにおいては、各クラスターのセクター数はディスクのサイズによって決定されます。DOS/Vは、ディスクのスペースを占有する8セクターのクラスターの数を計算して12ビットFATまたは16ビットのFATのどちらを作成するかを決めます。クラスターの数が4086以下であれば、12ビットのFATが作成されることになります。また4086以上であれば、16ビットのFATが作成されます。

ディスクのセクターの数は次の公式によって求めます。

TS=SPT × H × C

TS = ディスクの全セクターの数(Total number of sectors)
SPT = トラック当たりまたはシリンダー当たりのセクターの数
H = ヘッドの数(Heads)
C = シリンダーの数(Cylinders)

FATの最初の２つのエントリーはデータのマップ用には使用しません。これらのエントリーはディスクのサイズとフォーマットを表します。FATの最初のバイトは次のどれか１つを指定します。

16進値	**意味**
FF	40トラック両面、8セクター/トラックのディスケット(320KB)
FE	40トラック片面、8セクター/トラックのディスケット(160KB)
FD	40トラック両面、９セクター/トラックのディスケット(360KB)
FC	40トラック片面、９セクター/トラックのディスケット(180KB)
FB	80トラック両面、8セクター/トラックのディスケット(640KB)
FA	80トラック片面、8セクター/トラックのディスケット(320KB)
F9	80トラック両面、15セクター/トラックのディスケット(1.2MB)
F9	80トラック両面、９セクター/トラックのディスケット(720KB)
F8	ハード・ディスク
F0	その他

最初の２つのFATエントリーは、ディスクのサイズとフォーマットを表します。FATの第２および第３のバイトには値FFHが格納されます。第４のバイトは16ビットのFAT専用で、値FFHが格納されます。

32MB以上の媒体に対してDOS/Vがサポートする16ビットFATの最大サイズは、ディスク上の64KBまたは128KBのエントリー領域です。

ディスク・ディレクトリー

FORMATコマンドによって、ディスク全体のためのルート・ディレクトリーが作られます。ディスクのフォーマットが/Sオプションによって行われている場合は、DOS/Vのシステム・ファイル、(IBMBIO.COM, IBMDOS.COM, またはMSDOS.SYS, IO.SYSおよびCOMMAND.COM)がディスクに追加されます。次の８つのフォーマットは5.25インチのディスケットおよび3.5インチのディスケットに使用されているものです。

面数	セクター/トラック	FATサイズ・セクター	ディレクトリーのセクター	ディレクトリーのエントリー	セクター/クラスター
片面(5.25)	8	1	4	64	1
両面(5.25)	8	1	7	112	2
片面(5.25)	9	2	4	64	1
両面(5.25)	9	2	7	112	2
両面(5.25)	15	7	14	224	1
両面(3.5)	9	3	7	112	2
両面(3.5)	18	9	14	224	1
両面(3.5)	36	9	15	240	2

データ領域

データ・ファイルおよびサブディレクトリーはディスクの最後で、最大の部分に格納されます。スペースの割り振りは、必要に応じて１回に１クラスターずつ行われます。この割り振り方法であれば、ディスク空間を最も効率的に使用できます。クラスターが使用可能になると、新しいファイルに対するスペースの割り振りが行われます。

ディスクへのアクセス

ほとんどのディスクへのアクセスは21H割り込み機能を使って行えます。また他の５つの機能を使ってディスクに関連する作業を行うことができます。

機能	機能番号
ディスクのリセットおよびファイル・バッファーのフラッシング	0DH
省略時ディスク・ドライブの選択	0EH
現行ディスクへのアクセス	19H
ディスクの空き領域量のリクエスト	36H
開始時ドライブの決定	33H

ドライブおよびディスク情報のリクエスト

ディスクおよびドライブの情報は次の21H割り込み機能を使ってリクエストします。

機能	機能番号
現行ドライブ番号のリクエスト	19H
ディスク割り振り情報のリクエスト	1BH
指定したドライブに関するディスク割り振り情報のリクエスト	1CH

ディスクとの間のデータの直接読み取りおよび書き込み

DOS/Vでは、ディスクとの間の読み取りおよび書き込みに対して２つの割り込み、25Hと26Hが用意されています。

機能	割込番号
指定したディスク・セクターからの読み取り	25H
指定したディスク・セクターへの書き込み	26H

第3章　ファイル・ハンドルによるファイルへのアクセス

この章では次のタスクを完了するために必要な情報について説明します。

- ファイルに対するデータの読み取りと書き込み
- ファイル属性のリクエストと指定
- ディレクトリーへのアクセス
- ディレクトリー内のファイルの検索
- 各国語サポート(NLS: National Language Support)のリクエストと指定

ファイルの作成、オープン、クローズおよび削除のため、DOS/Vでは割り込み21H内で次の９つの機能を用意しています。

機能	機能番号
新しいファイルの作成または古いファイルの置き換え	3CH
ファイルのオープン	3DH
ファイル・ハンドルのクローズ	3EH
ファイルの削除	41H
ファイル名の変更	56H
新しいユニーク・ファイルの作成	5AH
新しいファイルの作成	5BH
ファイル領域への読み取りおよび書き込みアクセスのロックとロックの解除	5CH
拡張パラメーターによるファイルの作成およびオープン	6CH

ファイル名

ファイルに名前を付ける場合、アプリケーション・プログラムはそのファイルの名前と場所を示したASCIIZ文字列に対し、ポインターを与えます。ファイル名にはオプションのドライブ文字、パス、または16進０バイトで終了するファイルの仕様が含まれます。次にファイル名の文字列の例を示します。

```
'B:¥LEVEL1¥LEVEL2¥FILE1',0
```

ファイル名の最大サイズはドライブ、コロン(:)、パス、名前およびヌル・ターミネーターを含めて128バイトです。パス名を許すすべてのファンクション・コールは、パスの区切り文字としてスラッシュ(/)または円記号(￥)を受けつけます。

ファイル・ハンドル

オープンまたは作成のファンクション・コールはファイル・ハンドルと呼ばれる16ビットの値を返します。ファイルのI/O(入出力)を行うとき、プログラムはこのファイル・ハンドルを使ってファイルを参照します。ファイルが一度オープンされると、プログラムはそのファイルを指しているASCIIZの文字列を保存しておく必要はなくなります。どのディレクトリーが現行ディレクトリーになっていても、DOS/Vは対象ファイルの位置を見失いません。

機能	機能番号
ファイルに対する追加ファイル・ハンドルの指定(複写)	45H
既存ファイル・ハンドルで他のファイルを指す(強制複写)	46H
オープン・ファイル・ハンドル数の指定	67H

すべてのプロセスで一度にオープンできるファイル・ハンドルの数は、CONFIG.SYSのFILESコマンドで指定できます。１つのプロセスで使用できる省略時ハンドルは20です。なおプロセスが引き継いだハンドルはすべてリダイレクトさせることができます。

各オープン・ハンドルは１つのファイルまたは装置に対応することになっていますが、複数のハンドルが同じファイルまたは装置を参照することもできます。したがって、ハンドルの上限がFILESコマンドで指定された数を超える場合もあります。

特殊なファイル・ハンドル

DOS/Vにはアプリケーション・プログラムが使用する５つの特殊ファイル・ハンドルがあります。これらのハンドルは次のとおりです。

0000H標準入力装置(STDIN)

0001H標準出力装置(STDOUT)

0002H標準エラー装置(STDERR)

0003H標準補助装置(STDAUX)

0004H標準印刷装置(STDPRN)

各標準装置に対応するファイル・ハンドルはプログラムでオープンにする必要はありませんが、プログラムはそれらをクローズできます。STDIN は読取専用ファイルとして扱われます。STDOUTおよびSTDERRは書込専用のファイルとして扱います。なお

STDINとSTDOUTはリダイレクトさせることができます。標準装置へのアクセスはファンクション・コール01Hから0CHで行います。

標準装置のハンドルはコンソール装置との間で入出力を行う場合に便利です。たとえば、キーボードからの入力を読み取りファンクション・コール(3FH)およびファイル・ハンドル0000H(STDIN)を使用することで、読み取ることができますし、また書き込みファンクション・コール(40H)およびファイル・ハンドル0001H(STDOUT)で、コンソールの画面に出力を書き込むことができます。

STDOUTへの出力のリダイレクションが行われないようにするためには、ファイル・ハンドル0002H(STDERR)を使って出力を送ることで可能です。この機能はユーザーへのエラー・メッセージやプロンプトにも便利です。

ファイルとの間のデータの読み取りおよび書き込み

DOS/Vには、ファイルまたは装置との間の読み取りおよび書き込み、読み取りまたは書き込みが行われるファイルでのオフセットの指定、書き込み後読み取り状態(Read-after-write state)のベリファイを可能にする5つの機能があります。ただし、このベリファイによって動作が遅くなることがあります。

機能	機能番号
ファイルまたは装置からの読み取り	3FH
ファイルまたは装置への書き込み	40H
読み取りまたは書き込みが行われるアドレスの指定（ポインター使用）	42H
書き込み後読み取り状態の(Read-after-write state)リクエスト	54H
書き込み後読み取り状態の指定（ベリファイ・スイッチの設定と解除）	2EH

ファイル属性のリクエストと指定

ファイルを作成中でも、プログラムは特定の属性を指定できます。たとえば、作成の日付と時間、およびアクセスのレベルなどの指定が可能です。

機能	機能番号
ファイル属性のリクエストと指定	43H
ファイルの日付と時間のリクエストおよび指定	57H

サブディレクトリーへのアクセス

ルート・ディレクトリー以外のディレクトリー(サブディレクトリー)は、すべてファイルになります。物理的な媒体による収容が可能な限りは、サブディレクトリーに対するエントリー数の制限はありません。すべてのディレクトリー・エントリーは32バイト長になっています。

注: 値は16進数です。

ファイル名

ファイル名はバイト０から７で表します。ファイル名の第１のバイトでファイル名の状態が示されます。ファイル名の状態には次の値が格納されています。

00H 使用されたことがないファイル名。パフォーマンスの向上を図るときに、ディレクトリーの検索の長さを制限するために使用されます。

05H ファイル名の最初の文字がE5Hの文字です。

E5H ファイル名が使用されたが、ファイルは消去された状態です。

2EH ディレクトリー用のエントリーです。２番目のバイトも2EHであれば、クラスター・フィールドにこのディレクトリーの親ディレクトリーのクラスター番号が格納されます(親ディレクトリーがルート・ディレクトリーであればクラスター番号は0000H)。

この他の文字はファイル名の最初の文字になります。

注: バイトのオフセットは10進数です。

ファイル名の拡張

バイト８から10でファイル名の拡張を表します。

ファイル属性

バイト11でファイルの属性を表します。属性のバイトは次により割り当てられています。

01H 読み取り専用ファイルを表します。ファンクション・コール3DHまたは6CHを使って出力のためのファイルをオープンしようとすると、エラー・コードが返ります。

02H 隠しファイルを表します。このファイルは通常のディレクトリーの検索から除外されます。

04H システム・ファイルを表します。このファイルは通常のディレクトリーの検索から除外されます。

08H 最初の11バイトにボリューム・ラベルが格納されているエントリーを表します。エントリーにはその他の利用可能な情報は格納されておらず、ルート・ディレクトリーだけに存在します。

10H　エントリーはサブディレクトリーを定義することを示し、通常のディレクトリーの検索から除外されます。

20H　保存ビットを表します。このビットはファイルの書き込みが終わり、クローズされるとONにセットされます。これは作成されてから、または前回の更新以後、ファイルが変更になっているかどうか確認するときに、BACKUPおよびRESTOREコマンドによって使用されます。なお、このビットは他の属性ビットと一緒に使用することもできます。

他のビットはすべて予約済みで０でなければばなりません。

ファイルの作成時刻と最新の変更時刻

バイト22と23にはファイルが作成された時刻、または最後に更新になった時刻が格納されています。この時刻は次のようにビットに割り当てられます。

```
<          23           > <           22          >
15 14 13 12 11 10 9  8  7  6  5  4  3  2  1  0
h  h  h  h  h  m  m  m  m  m  m  x  x  x  x  x
```

この場合

hh = 時、２進数(0-23)
mm = 分、２進数(0-59)
xx = ２秒ずつ増加、２進数

時刻に関しては、下位のバイトが最初に格納されます。

ファイル作成日付

バイト24と25にファイルの作成または最後に更新された日付が格納されています。この場合、mm/dd/yyが次のビットに割り当てられます。

```
<          25           > <            24         >
15 14 13 12 11 10 9  8  7  6  5  4  3  2  1  0
y  y  y  y  y  y  y  m  m  m  m  d  d  d  d  d
```

ただし

mm = 1–12
dd = 1–31
yy = 0–119(1980–2099)

日付に関しては、下位バイトが最初に格納されます。

開始クラスター番号

バイト26と27にファイルの最初のクラスターのクラスター番号が格納されています。すべてのハード・ディスクおよびディスケットのデータ領域の最初のクラスターは、クラスター002です。クラスター番号は、下位バイトが最初に格納されます。

```
<        27          >  <          26         >
  0  0  0  0  0  00  0  0  0  1  0  0  0  0  1
```

ファイルのサイズ

バイト28から31までにバイトによるファイルのサイズが格納されています。最初のワードにはサイズの下位部分が格納されます。両方のワードともに下位バイトが最初に格納されます。

ディレクトリーへのアクセス

DOS/Vにはディレクトリーの作成、移動、変更または削除のために割り込み21H内で使用できる４つの機能が用意されています。

機能	機能番号
サブディレクトリーの削除	3AH
サブディレクトリーの作成	39H
他のディレクトリーへの変更	3BH
現行ディレクトリーの識別	47H

ディレクトリーのファイルの検索

DOS/Vには最初に一致したエントリーと次に一致したエントリーを検索するために、割り込み21H内で使用できる２つの機能が用意されています。

機能	機能番号
最初に一致するエントリーの検索	4EH
次に一致するエントリーの検索	4FH

各国語サポート(NLS)のリクエストと指定

DOS/VにはNLSに対し、次の機能が用意されています。

機能	機能番号
現在の国別情報指定	38H
国別依存情報のリクエスト	38H
２バイト文字セット(DBCS)のサポート	65H

ネットワーク操作の制御

ネットワークがロードされている場合、DOS/Vのいくつかのファンクション・コールが入力としてネットワーク・パスを受けつけます。ネットワークへのアクセスが可能であれば、さらに情報がB-1ページの付録B、『DOS/Vファンクション・コール』の各関連ファンクション・コールの「備考」欄に示されています。ネットワーク・パスは、コンピューター名、ディレクトリー・パス、およびオプションのファイル名を含んだASCII文字列で構成されています。ネットワーク・パスにドライブ指定子を含めることはできません。このパスは２進数の０バイトで終了させます。次に例を示します。

```
¥¥SERVER1¥LEVEL1¥LEVEL2¥FILE1
```

入力としてASCIIZ文字列を受けつける、ほとんどのファンクション・コールは、ネットワーク・パスも受けつけます。たとえば、5BHの機能(新しいファイルの作成)を実行しようとする場合、「読み取り／書き込み／作成」または「書き込み／作成」でディレクトリーにアクセスして、ファイルを作成しなければなりません。「読み取り専用」または「書き込み専用」のアクセスが可能であって、「作成」のアクセスが不可能なときは、そのディレクトリーでファイルを作成することはできません。入力としてネットワーク・パスを受けつけないファンクション・コールは「現行ディレクトリーの変更(3BH)」と「最初に一致するファイルの検索(4EH)」の２つです。

ネットワークの操作制御に使用できるファンクション・コールは次のとおりです。

機能	機能番号
ファイル領域への読み取りまたは書き込みアクセスの、ロックとロックの解除	5CH
ファイルから装置に全データを書き込む	68H
ローカル・コンピューターIDのリクエスト	5E00H
プリンター・セットアップ文字列の指定	5E02H
プリンター・セットアップ文字列のリクエスト	5E03H
リダイレクションのリクエスト	5F02H
リダイレクト装置への接続	5F03H
リダイレクションの取消	5F04H

第4章　ファイル制御ブロックを使用したファイルへのアクセス

この章では次のタスクの実行に役立つガイドとシステム・アーキテクチャーに関する情報について説明します。

- ファイルへのアクセス
- 順次レコードへのアクセス
- ランダム・レコードへのアクセス
- ディレクトリー内のファイルの検索
- ドライブおよびディスク情報のリクエスト

ファイル制御ブロック(FCB)

わずかな例外を除いて、ファイル制御ブロック(FCB: File Control Blook)を使用してファイルを維持しなければならないのは、プログラムをDOS 1.10で実行させる場合だけです。FCBを使用すると、ユーザーのプログラムは00Hから2EHまでのファンクション・コールしか利用できなくなります。ファイルのアクセスにはファイル・ハンドルの使用を推奨します。

オープンしたファイルごとに、１つのFCBを、ユーザーのプログラムとDOS/Vによって維持する必要があります。FCBに対しては、ユーザーのプログラムからポインターを与え、特定のファンクション・コールによって、リクエストされているフィールドを埋めます。

FCBの予約フィールドを、プログラムが使用しないように注意してください。バイト０から15、およびバイト32から36までは、ユーザーのプログラムで設定しなければなりません。バイト16から31はDOS/Vで設定し、ユーザー・プログラムによって変更されないようにしなければなりません。

オープンされていないFCBは、FCBのプリフィックス(使われている場合)、ドライブ番号、ファイル名、指定されたエクステンションによって構成されています。オープンされているFCBでは、作成またはオープンのファンクション・コールによって指定した残りのフィールドです。

すべてのワード・フィールドは、下位バイト(LSB)から最初に格納されます。たとえば、128のレコード長はオフセット14の80H、およびオフセット15の00Hとして格納されます。詳細は4-2ページの図4-1を参照してください。

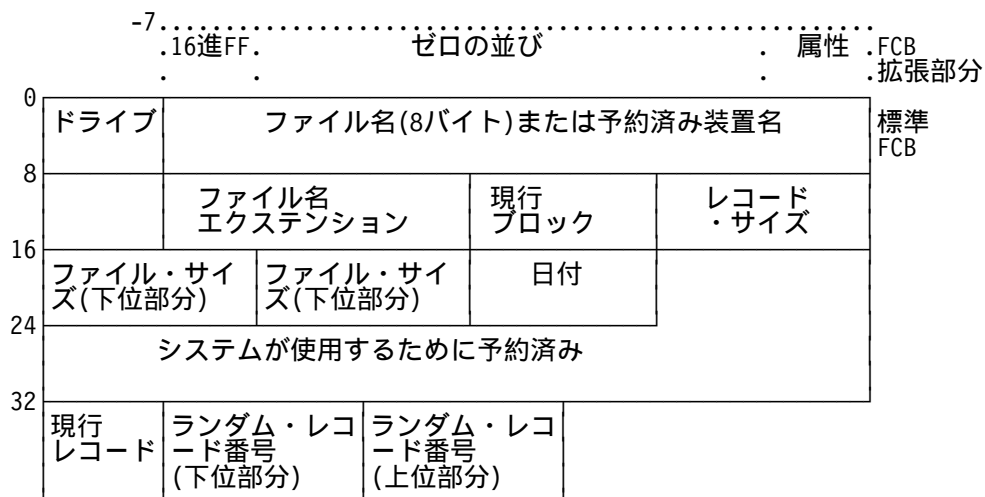



図　4-1. ファイル制御ブロック

ファイル・サイズ、日付、システムが使用するため予約済みの部分はDOSが設定しますから、値を変更しないようにしてください。それ以外の部分はFCBを使うプログラムが値を設定しなければなりません。

注:　オフセットは10進数。

FCBは次のような形式です。

ドライブ番号

バイト０はドライブ番号を表します。オープンする前は０は省略時ドライブ、１はドライブA、２はドライブBと等しくなります。オープンすると、０がドライブA、１がドライブA、２がドライブBと等しくなります。

なお、ファイルがオープンされると、ドライブ番号０は、実際のドライブ番号で置き換えられます。

ファイル名

バイト１から８でファイル名を表し、左寄せにして残りにスペースが入ります。ここでLPT1のような予約装置名を指定するときは、コロン(:)を含めないでください。

ファイル名のエクステンション

バイト９から11でファイル名のエクステンションを表し、残りまたは全体に空白が入ります。

現行ブロック番号

バイト12と13は現行ブロック番号を表わし、この番号はファイルの先頭の相対値で０で始まります。０はオープン・ファンクション・コールで設定します。１つのブロックは128のレコードで構成され、それぞれのサイズは論理レコード・サイズのフィールドで指定されます。現行ブロック番号と現行レコード・フィールドは、順次読み取りおよび書き込みに使用します。

論理レコードのサイズ

バイト14と15で論理レコード・サイズをバイトで表します。80Hはオープンのファンクション・コールで設定します。論理レコード・サイズを80Hから変更するときは、この値をリセットできます。DOS/Vではこの値を使って全ディスクの読み取りと書き込みにおいて、ファイル中の位置を決めるのに使用します。

ファイル・サイズ

バイト16から19では、ファイルのサイズをバイトで表します。この2ワードのフィールドでは、最初のワードがサイズの下位部分になります。

ファイル・データ

バイト20と21でファイルの作成日付、または最後の更新日付を表します。*mm/dd/yy*はつぎのようにビットに割り当てられています。

```
<         21         > <         20         >
15 14 13 12 11 10 9  8  7  6  5  4  3  2  1  0
y  y  y  y  y  y  y  m  m  m  m  d  d  d  d  d
```

この場合

*mm*は1-12
*dd*は1-31
*yy*は0-119(1980-2099)

予約済み

バイト21から31までは予約済みです。

ブロックのレコード番号

バイト３は、現行ブロック内で現行相対レコード番号(0-127)を表します。このフィールドは、ディスケットに対する順次読み取りおよび書き込みの作業の前に、設定しておく必要があります。このフィールドは、オープンのファンクション・コールで初期化はされません。

ファイル内のレコード番号

バイト33から36はレコード番号を表し、この番号はファイルの先頭の相対値で０で始まります。このフィールドは、ディスケットに対するランダム・読み取り、および書き込み作業の前に設定しておく必要があります。このフィールドは、オープンのファンクション・コールで初期化はされません。

レコード・サイズが64バイトより小さいときは、両方のワードを使用します。またレコード・サイズが64バイトよりも大きければ、最初の3バイトだけ使用します。プログラム・セグメントの5CHのFCBを使用すると、FCBの最後のバイトと、未フォーマットのパラメーター領域の最初のバイトが重なりますから、注意してください。

拡張FCB

拡張FCBは、特殊な属性のファイルをディスク・ディレクトリー内で作成、または検索するときに使用します。拡張FCBでは次のような７バイトのプリフィックスが追加されています。次のようにフォーマットされます。

拡張FCB

FCBのバイト-7には拡張FCBを表すFFHが格納されます。

予約済み

FCBのバイト-6から-2は予約済みです。

ファイル属性

FCBのバイト-1は属性バイトを表します。ファンクション・コール00Hから2EHは標準FCBと拡張FCBの両方に対して有効です。拡張FCBを使用するときは、適切なレジスターをドライブ番号フィールドではなくて、プリフィックスの最初のバイトに設定します。

ディスク転送領域(DTA)

DOS/Vではメモリー中のバッファーを、ディスク転送領域(DTA: Disk Transfer Area)として使用し、FCBファイルの読み取り、および書き込みのデータを保持します。このDTAはユーザー・プログラムのデータ領域内のどの位置にあっても構いません。またユーザーのプログラムによって指定します。

有効なDTAは、１回に１つだけしかありませんから、ディスク読み取りまたは書き込み機能を発行する前に、どのメモリー位置を使用するか、ユーザー・プログラムでDOS/Vに指示しなればなりません。プログラムの制御がCOMMAND.COMによって行われるときは、プログラム・セグメント・プリフィックスの80Hに、128バイトを保持できるだけの省略時DTAが設定されます。

DOS/Vでは、DTAの動作を処理するための、次の機能が割り込み21Hに用意されています。

機能	機能番号
DTA内でのデータ読み取り、および書き込みのためのバッファー・アドレスの指定	1AH
DTA内でのデータ読み取り、および書き込みのためのバッファー・アドレスのリクエスト	2FH

ファイル内でのアクセス

FCBは、指定ドライブの現行ディレクトリーだけでなく、すべての有効なドライブのファイルを識別することができます。

SHAREがロードされていない場合、１回にオープンできるファイルの数(FCBファンクション・コールの使用による)に制限はありません。しかし、ファイル共用がロードされている場合には、FCBのオープン・ファイルの最大数は、CONFIG.SYS内のFCBSコマンドで指定されている値によって制限されます。

ユーザーはFCBSコマンドを使って、mの値を指定できます。mの値は、一度にオープンできるFCBの数です。

FCBオープンの最大数を超えたときは、最も利用されていないファイルを、DOS/Vが自動的にクローズします。このようなファイルにアクセスしようとすると、割り込み24Hで重大エラー・メッセージ、「FCBがありません」が現れます。プログラムの実行中にこのような状態になった場合、FCBSコマンドのmで指定した値を、大きくする必要があります。

最初のオープン・ファイルをクローズせずに、同一のFCBを、第二のファイルをオープンするのに使用しないでください。同時に複数のファイルをオープンするときには、別のFCBを使用します。ファイル共用による潜在的な問題を回避するためには、入出力が行われた後にファイルをクローズします。なお、ファイルの削除または名前の変更は、ファイルをクローズしてから行ってください。

FCBSコマンドの使用によるファイルの管理は次のファンクション・コールを使って行います。

機能	機能番号
ファイルのオープン	0FH
ファイルのクローズ	10H
ファイルの削除	13H
ファイルの作成	16H
ファイル名の変更	17H
ファイル・サイズのリクエスト	23H
ファイル名の情報を構成要素に分ける(構文解析)	29H

順次レコードのアクセス

現行ブロック、現行レコード、およびFCBのレコード長フィールドを使用することで、ユーザーは順次入出力を行うことができます。この場合、次の割り込み21Hの順次読み取り、および書き込み機能を使用します。

機能	機能番号
レコードの読み取り	14H
レコードの書き込み	15H

ランダム・レコードのアクセス

ランダム入出力は、FCBのランダム・レコードとレコード長フィールドを埋め、割り込み21Hの、以下のファンクション・コールを行うことで実行できます。

機能	機能番号
シングル・レコードの読み取り	21H
シングル・レコードの書き込み	22H
FCBのランダム・レコード・フィールドの指定	24H
複数レコードの読み取り	27H
複数レコードの書き込み	28H

ディレクトリー内のファイル検索

FCBをソースとして使用し、ディレクトリー内のファイルの検索や変更を行うには、割り込み21Hの、以下の機能で行います。

機能	機能番号
最初に一致したファイル・エントリーの検索	11H
次に一致したファイル・エントリーの検索	12H
ファイルの作成	16H
ファイルの削除	13H

機能	機能番号
ファイル名の変更	17H
ファイル名の情報を構成要素に分ける(構文解析)	29H

第5章　装置入出力の管理

この章では次のタスクに関するガイド、およびシステム・アーキテクチャーの情報について説明します。

- ディスプレイ入出力の管理
- キーボード入出力の管理
- その他の入出力の管理
- 装置の転送管理
- システム・デバイス・ドライバーの制御チャネルのアクセス

ディスプレイ入出力の管理

DOS/Vでは画面に文字または文字列を送る、４つの機能が割り込み21Hで用意されています。

機能	機能番号
画面への一文字出力（Control-Break割り込みハンドラーのトリガー可）	02H
一文字入力待ちおよび画面への出力(Control-Break割り込みハンドラーのトリガー不可)	06H
メモリーの文字列を画面に出力	09H
バッファーの文字列を画面に出力、または文字列のファイル装置への書き込み	40H

文字属性の指定、画面の前景および背景色、およびANSI.SYSを使用する画面サイズの指定の詳細は、F-1ページの付録F、『ANSI.SYS（キーボード／画面の拡張制御）』を参照してください。

キーボード入出力の管理

DOS/Vでは、ユーザーのアプリケーション・プログラムで、キーボード入出力を管理するため、完全な補助機能が割り込み21Hで用意されています。

機能	機能番号
キーボードからディスプレイへの入力を送る(エコー付き)	01H
キーボードから入力を直接受け取る、または出力をディスプレイに直接送る	06H
キーボードから入力を直接エコーなしで受け取る	07H
キーボードからの入力をディスプレイへのエコーなしで受け取る(Control-Break割り込みハンドラーのトリガー可)	08H
キーボードからバッファーへの文字の読み取り	0AH
キーボード・バッファーの状態のチェック	0BH
キーボードバッファーのクリア、バッファーをクリアした後、コールする機能を指定	0CH

その他の入出力の管理

その他の入出力の管理には次の３つの機能の使用が可能です。

機能	機能番号
補助入力	03H
補助出力	04H
プリンター出力	05H

ファイル・システム動作の管理

次に示すシステム動作をDOS/Vはサポートしています。

機能	機能番号
ローカル・コンピューターIDのリクエスト	5E00H
プリンター・セットアップ文字列の指定	5E02H
プリンター・セットアップ文字列のリクエスト	5E03H
リダイレクション・リストのリクエスト	5F02H
リダイレクション装置への接続	5F03H
リダイレクションの取消	5F04H
ファイルから装置への全データの書き込み	68H

システム・デバイス・ドライバーの制御チャネルへのアクセス

割り込み21Hの機能番号44Hは、デバイス・ドライバー制御チャネルをアクセスするための、多目的なファンクションです。44Hを使用することによって、ユーザーのアプリケーション・プログラムは、装置の状態についてリクエストしたり、入出力制御チャネルへの読み取り、および書き込みをリクエストできます。次のサブファンクションの値を、ALに入れて渡します。

カテゴリー	機能	サブファンクション番号
装置情報のリクエストおよび指定	装置情報のリクエスト	00H
	装置情報の指定	01H
	装置に、取り外し可能な媒体が含まれているかどうかの確認	08H
キャラクター装置からの読み取り、および書き込み	キャラクター装置からの読み取り	02H
	キャラクター装置への書き込み	03H
ブロック装置からの読み取り、および書き込み	ブロック装置からの読み取り	04H
	ブロック装置への書き込み	05H
論理ドライブのリクエストおよび指定	論理ドライブのリクエスト	0EH
	論理ドライブの指定	0FH
装置に対するネットワーク・サポートの供給	DOS/V共用ファイルの矛盾の解決を試す回数（および間隔）の指定	0BH
	ファイル・ハンドルに対する入出力制御	0CH
	ブロック装置に対する入出力制御	0DH
	論理装置がローカルか、リモートかの決定	09H
	ファイル・ハンドルがローカルか、リモートかの決定	0AH

バイナリーおよびASCIIモードでのデータの読み取りおよび書き込み

装置へのデータの読み取り、または書き込みのモードを変更するときは、プログラムは機能番号44Hを使用します。バイナリー・モードで入出力が行われる場合は、制御値には意味はありません。ASCIIモードで入出力が行われるときには、特定の制御値は意味を持ちます。それらは以下の表に示します。

制御値	キーボード入力	意味
1AH	∧Z	ファイルの末尾(End Of File)
0DH	∧M	復帰(Carriage Return)
0AH	∧J	改行(Line Feed)
03H	∧C	制御ブレイク(Control Break)
13H	∧S	スクロール・ロック(Scroll lock)
10H	∧P	両面印刷(Print Screen)
11H	∧Q	スクロール再スタート(Scroll restart)
04H	∧D	タスクの終り(End of Task)

ASCIIモードでファイルを読み取ると、そのファイルがディスプレイにエコーされ、タブはスペースに拡張されます。これは入力バッファーにタブ・バイト(09H)として残されます。またASCIIモードでファイルを書き込むと、タブは８文字の境界まで拡張され、スペース(20H)で埋まります。

第6章　プロセス制御

この章では次の動作に関するガイドと、システム・アーキテクチャーについて説明します。

- プログラムのロード時の識別
- サブプログラムのロードと実行
- プログラムまたはサブプログラムの終了
- 実行なしのオーバーレイのロード
- コマンド・プロセッサーのコール
- エラーに対する対応
- 制御ブロックの動作に対する対応
- システムの日付、時間のリクエストと指定
- 割り込みベクトルのリクエストと指定

メモリーの割り当て

DOS/Vは割り当てられ、使用が可能になったメモリー・ブロックを管理するとともに、アプリケーション・プログラムがそのメモリー・リクエストを伝えるため、３つのファンクション・コールが用意されています。

機能	機能番号
メモリーの割り当て	48H
割り当てられたメモリーの解放	49H
割り当てられたメモリー・ブロック・サイズの変更	4AH

DOS/Vのメモリー管理

DOS/Vは、「パラグラフ」と呼ばれる16バイトのユニットを割り当て、割り当てられた各ブロックに対し、「制御ブロック」を作ることで、メモリーの管理を行います。この割り当ては、実際のリクエストよりも16バイト大きくなります。それはDOS/Vが各割り当てブロックを管理するため、一つの「制御ブロック」を自動的に割り当てるからです。

ユーザーがコマンド・ラインでプログラムを開始すると、COMMAND.COMが実行可能なプログラム・モジュールを利用可能メモリーの最大の未使用ブロックにロードし、ファイル・ヘッダーを読み込みます。

利用可能なメモリーが不十分なときは、システムがエラー・コードを返し、制御をプログラムに移します。この場合、ユーザーのプログラムでは割り当てられたメモリーを、

必要なサイズだけ小さくするには、SETBLOCKファンクション・コール(4AH)を使用します。

注: DOS/Vが提供した省略時スタックが、解放されたメモリー領域に存在することが考えられます。したがって、SETBLOCKを発行する前に、独自のスタックを設定するように、.COMプログラムに覚えておく必要があります。SETBLOCKコールは、必要とされていないメモリーを解放し、以後のプログラムのロードに使用できるようにします。

プロセスの最中に、ユーザーのプログラムが追加のメモリーを必要とする場合は、割り込み21の中のファンクション・コール48Hを発行します。メモリーを解放にするには、割り込み21内のファンクション・コール49Hを発行します。

DOS/Vのメモリー・マップ

次の表は、DOS/Vが基本メモリーにロードされた場合に、DOS/Vの構成要素およびアプリケーション・プログラムが、どのような順序でロードされるかを示しています。

位置	用途
0000:0000	割り込みベクトル・テーブル
0040:0000	ROMコミュニケーション領域
0050:0000	DOS/Vコミュニケーション領域
XXXX:0000	IBMBIO.COMまたはIO.SYS – DOS/VのROM入出ルーチン・インターフェース
XXXX:0000	IBMDOS.COMまたはMSDOS.SYS – DOS/Vの割り込みハンドラーとサービス・ルーチン（INT 21機能）
XXXX:0000	DOS/Vのバッファー、制御領域、および導入されたデバイス・ドライバー
XXXX:0000	COMMAND.COMの常駐部分 – 割り込み22H (終了)、23H (Control Break)、24H (重大エラー)の割り込みハンドラーと一時的部分を再ロードするコード
XXXX:0000	外部コマンドまたはユーティリティー.COMまたは.EXEファイル
XXXX:0000	.COMファイル用ユーザー・スタック
XXXX:0000	COMMAND.COMの一時的な部分

次の表は、DOS/VがメモリーのHMA領域にロードされた場合に、DOS/Vの構成要素およびアプリケーション・プログラムが、どのような順序でロードされるかを示しています。

位置	用途
0000:0000	割り込みベクトル・テーブル
0040:0000	ROMコミュニケーション領域
0050:0000	DOS/Vコミュニケーション領域
0070:0000	常駐BIOSデータ–デバイス・ドライバー・ハンドラーとそのエントリー・ポイントを含む。
XXXX:0000	DOS/Vデータ
XXXX:0000	DOS/V導入可能デバイス・ドライバー、およびSYSINTによって割り振られたデータ構造
FFFF:0010	VDISKヘッダー
FFFF:0030	IBMBIOまたはIO.SYS
XXXX:XXXX	IBMDOSまたはMSDOS.SYS

メモリー・マップ・アドレスはセグメント・オフセット形式です。たとえば、0070:0000は絶対アドレス00700Hです。

注: VDISKヘッダーはHMAの先頭に置かれます。INT 15Hメモリーを使用するアプリケーションは、VDISKヘッダーをチェックしてVDISKヘッダーがあった場合はHMAを使用しません。

DOS/Vのコミュニケーション領域は次のように使用されます。

0050:0000 画面印刷状態フラグの格納

0	非活動状態画面印刷または画面印刷作業完了
1	進行中の画面印刷
255	画面印刷中にエラーが発生

0050:0004 単一ドライブ・モードの状況バイト

0	ドライブAのディスケットが最後に使用された
1	ドライブBのディスケットが最後に使用された

0050:0022—002F ディスケットの初期化のためDOS/Vが使用

0050:0030—0033 MODEコマンドが使用

0050:0000で始まっている256バイト以内のその他の場所はDOS/V用として予約されています。

ロード時のプログラムの識別

DOS/Vには、アプリケーション・プログラムに対し、ロード時にプログラム自体を指定または識別するファンクション・コールが２つ用意されています。

機能	**機能番号**
ロード時にプログラム・セグメント・プリフィックス(PSP)を通じてDOS/Vがプログラムを識別する手段の設定	26H
ロード時にDOS/Vが識別する方法のリクエスト	62H

プログラム・セグメント

ユーザーが外部コマンドを入力したり、EXECファンクション・コール(4BH)でプログラムをコールすると、DOS/Vはメモリー中の利用可能な最下位アドレスを決定し、それをプログラムに割り当てます。このメモリー領域は「プログラム・セグメント」と呼ばれます。DOS/Vはプログラム・セグメントのオフセット０のところに「プログラム・セグメント・プリフィックス」制御ブロックを作ります。EXECが発行されると、DOS/Vはオフセット100Hにプログラムをロードし、制御を渡します。プログラム・セグメント・プリフィックスについては図6-1を参照してください。

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>A</td><td>B</td><td>C</td><td>D</td><td>E</td><td>F</td></tr>
<tr><td colspan="2">INT 21H</td><td colspan="2">メモリーの最上位</td><td colspan="6">予約済み</td><td colspan="2">終了アドレス IP</td><td colspan="2">終了アドレス IP CS</td><td colspan="2">CtrlBreak出口アドレス IP</td></tr>
<tr><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td colspan="6">16 - 2B</td><td>2C</td><td>2D</td><td colspan="2">2E - 4F</td></tr>
<tr><td colspan="2">CtrlBreak出口アドレス CS</td><td colspan="4">重大エラー出口アドレス</td><td colspan="6">予約済み</td><td colspan="2">環境ポインタ</td><td colspan="2">予約済み</td></tr>
<tr><td>50</td><td>51</td><td colspan="2">52 - 5B</td><td colspan="8">5C - 6B</td><td colspan="4">6C - 7F</td></tr>
<tr><td colspan="2">DOSコール</td><td colspan="2">予約済み</td><td colspan="8">非オープン標準FCB1</td><td colspan="4">非オープン標準FCB2</td></tr>
<tr><td>80</td><td colspan="15">81 - FF</td></tr>
<tr><td>Parm長さ</td><td colspan="15">先頭がブランクで始まるコマンド・パラメーター</td></tr>
</table>

図　6-1. プログラム・セグメント・プリフィックス

プログラム・セグメント・プリフィックスの、利用可能メモリーの最初のセグメントは、パラグラフの形式になっています。すなわち、1000Hが64KBを表します。オフセット６のワードには、セグメントで利用が可能な、バイトの番号が格納されています。

オフセット2CHには環境のパラグラフ・アドレスです。

オフセット50Hには、DOS/Vのファンクション・ディスパッチャーを呼ぶコードが含まれています。必要な機能番号をAHに置くことによって、プログラムがPSP＋50Hに対してlongコールを行い、割り込み21Hを発行せずにDOS/Vの機能をコールすることができます。

省略時のディスク転送アドレスは80Hに設定されます。

81Hの未フォーマット・パラメーター領域には、コマンド名の後に入力された、すべての文字が格納されています。これには先行または埋め込みのデリミタ(区切り文字)が含まれ、80Hは文字数を設定します。コマンド行に<、>、またはパラメーターが入力されると、これらのパラメーターとそれに対応するファイル名は、この領域に現れません。なぜなら、標準入力および出力のリダイレクションを行っても、アプリケーションからは見えないからです。

.ファイルの場合、オフセット６(１ワード)に、そのセグメントで利用が可能なバイト数が格納されます。

レジスターAXは、最初２つのパラメーターを、以下のように入力したドライブ指定子が格納されます。

AL=最初のパラメーターに無効なドライブ指定子が含まれていれば、FFH(それ以外はAL=00H)
AH=第２のパラメーターに無効なドライブ指定子が含まれていれば、FFH(それ以外はAH=00H)

.EXEプログラムでは、DSおよびESレジスターが、プログラム・セグメントに対するポインターを設定し、CS、IP、SS、およびSPレジスターは、リンカーから渡された値に設定されます。

.COMプログラムでは、プログラム・セグメント・プリフィックス・制御ブロックで始まる、初期割り当てブロックのセグメント・アドレスを、４つのセグメント・レジスター全部が格納します。インストラクション・ポインター(IP)は100Hに設定されます。オフセット６セグメント・サイズは、パラグラフのサイズに合わせて「丸め」が行われます。

オーバーレイのロードと実行

ユーザーのプログラムで、オプションのオーバーレイをロードするときには、4BHファンクション・コールを使用します。機能4BH、値０(AL=0)は、オーバーレイ付きのプログラムをロードし、実行します。また機能4BH、値３(AL=3)は、オーバーレイを実行せずロードだけを行います。

ユーザーのプログラムがオーバーレイをコールすると、EXECコールは呼び出しているプログラムがオーバーレイのために、すでにメモリーを割り当て済みであるものとみなします。オーバーレイのロード・リクエストでは、該当のオーバーレイをロードするメモリーを、呼び出しているプログラムが持っているかどうかの確認をしません。割り当てられていないメモリーにオーバーレイがロードされると、DOS/Vメモリー管理制御ブロックが破壊されます。これはDOS/Vが連続して制御ブロックを使用する必要がない限り、明らかになりません。

メモリーの割り当てエラーが返ってきたら、問題を訂正し、システムを再始動させなければなりません。オーバーレイは、自分が作動するメモリーを持っていないため、SETBLOCKコールを発行すべきではありません。メモリーはコールプログラムによって制御されます。

パラメーター・ブロック

ユーザーのプログラムが、EXECコール(4BH)を使ってサブプログラムをコールすると、サブプログラムに以下のようなものを供給する、パラメーター・ブロックを渡すことができます。

- 環境文字列
- 他のコマンド・プロセッサーと同じように動作をするコマンド行
- プログラム・セグメント・プリフィックスの5Cおよび6Cのファイル・制御ブロック(オプション)

環境文字列

呼び出しているプログラムから渡された環境は、そのプログラムの環境のコピーです。渡された環境のセグメント・アドレスは、プログラム・セグメント・プリフィックスのオフセット2CHに格納されます。

環境は一連のASCII文字列で、合計は32KB以下の次のような形式です。

NAME=parameter

注:　「NAME=」は必ず大文字です。

各文字列は０のバイトで終わります。また、すべての一連の文字列は、別な０のバイトで終ります。別のASCII文字列はワード・カウントを格納し、またASCIIZ文字列は一連の環境文字列に続き、実行可能なプログラムのドライブ、パス、ファイル名、およびエクステンションを格納しています。

コマンド・プロセッサーが構築され、すべての被コールプログラムに引き渡された環境には、COMPSEC= *string*、発行された最後のPATH、APPENDおよびPROMPTコマンド、およびSETコマンドで指定された環境文字列が格納されます。

コマンド・ライン

ユーザー・プログラムでは、サブプログラムに移行されるコマンド行を作成しなければなりません。

ファイル制御ブロック

ファイル・ハンドルを基本にしているファイルを、ユーザー・プログラムで使用する場合は、ファイル制御ブロックには関係ありません。ユーザー・プログラムでファイル制御ブロックを使用し、5CHか6CHにパス名が格納されているときは、対応するFCBは有効なドライブ番号しか格納しません。またファイル名フィールドも有効でなくなります。

プログラムまたはサブプログラムの終了

DOS/Vには、プログラムを終了させるために、４つの機能と２つの割り込みが用意されています。また終了にともなって、どこに制御を引き渡すかを指定するための、割り込みも用意されています。

機能	機能番号
プログラムを終了させて、制御をコールプロセスに渡す	4CH
プログラムを終了させて、指定部分をメモリーに残す	31H
プログラムの終了	00H
プロセス終了方法の決定	4DH

割り込み20Hはプログラムを終了します。割り込み27Hは、メモリーに指定部分を残し、プログラムを終了します。割り込み22Hは、プログラムの終了にともなって、制御をどこに渡すかを指定します。

サブプログラムが終了すると、コールプログラムに制御が戻されます。終了する前に、サブプログラムに割り当てたメモリーを、呼び出しているプログラムがシステムに戻さなければなりません。　呼び出しているプログラムが終了すると、DOS/Vに制御が戻されます。DOS/VはCOMMAND.COMの常駐部分が変更になったかどうか確認のため、CHECKSUMをします。変更になっていれば、環境で指定されているパスに基づいて、DOS/VがCOMMAND.COMを再ロードします。

プログラムは次のどれか１つの方法によって実行から戻されます。

- プログラム・セグメント・プリフィックスのオフセット０へのジャンプ
- INT 20Hの発行

- レジスターAH=00Hまたは4CHで、INT 21Hを発行
- AH=00Hまたは4CHで、プログラム・セグメント・プリフィックスの位置50Hをコールする

  INT 21Hの使用をお勧めします。

  4CHのコールを除いて、前述のいずれかの方法で終了させる場合、プログラム・セグメント・プリフィックスのセグメント・アドレスをCSレジスターが格納しているかどうか、すべてのプログラムで確認しなければなりません。

  前述のすべての方法が、EXECを発行したプログラムに制御を返すことになっています。プロセスの間に、割り込みベクトル22H、23H、および24H(終了、Ctrl-Break、および重大エラーの出口アドレス)が、終了プログラムのプログラム・セグメント・プリフィックスに保存されている値から復元されます。次に終了アドレスに制御が与えられます。

オーバーレイを実行せずにロードする

ファンクション・コール4BHでAL=３が指定されると、プログラム・セグメント・プリフィックスは造られず、DOS/Vは呼び出しているプログラムがオーバーレイに対してメモリーを割り当てたものとみなします。呼び出しているプログラムは次のどちらかの方法でメモリーを供給します。

- SETBLOCKコール(4AH)を発行したときに、オーバーレイに十分なメモリーを供給する
- 49Hコールでメモリーを供給する

DOS/VがAL=３のリクエストを受け取ると、システムはリクエストされたメモリーをコールプログラムが、保有しているものとみなします。サブプログラムと同じように、割り当てられていないメモリーであっても、オーバーレイをロードすることができますが、このようなときには、一連のDOS/Vメモリー管理制御ブロックを破壊します。

AL=３でロードしたプログラムは、SETBLOCKコール(4AH)を発行すべきではありません。これは動作をするのに必要なメモリーがオーバーレイではなく、呼び出しているプロセスによって保有されているためです。終了の前に、呼び出しているプログラムはサブプログラムに割り当てたメモリーを、システムに返さなければなりません。呼び出しているプログラムが終了すると、制御がDOS/Vに戻されます。

コマンド・プロセッサーのコール

コマンド・プロセッサーをコールするには、次の方法によって行なわなければなりません。

- コマンド・プロセッサーの第２コピーおよびその実行するコマンドを格納しておくために、適切な空きメモリーがあるかどうかを保証する。4AHファンクション・コールを発行し、ユーザーの現在の必要に合わせ、割り当てたメモリーを縮小す

る。BX=FFFFHで48Hファンクション・コールを発行する。返ってくるのが利用可能なメモリー。

- 次の形式で第２コマンド・プロセッサーのパラメーター文字列を作る。
  1 byte = パラメーター文字列の長さ
  xx byte = パラメーター文字列
  1 byte = 0DH(復帰)

  次のアセンブラのステートメントは、DISKCOPYコマンドを実行する列。

```
DB 19, "/C C:DISKCOPY A: B:" , 13
```

- EXECファンクション・コール(4BH、機能値０)を使ってコマンド・プロセッサーの第２コピーを実行する。PSP+2CHで渡された環境のパラメーターCOMSPEC=は、ドライブ、ディレクトリー、およびコマンド・プロセッサーの名前を示す。オフセット２をEXEC制御ブロックに設定して、必ずパラメーター文字列を指しておくこと。

エラーの対応

DOS/Vの機能を実行できないときは(重大エラーの状態を表します)、制御が割り込み24Hに移ります。機能59Hによって追加のエラー状況が得られます。

機能	**番号**
重大エラーの対応	割り込み24H
追加のエラー情報および対処方法のリクエスト	機能59H

ハンドル・ファンクション・コールはキャリー・フラグをセットし、エラー・コードをAXに返すことで、エラーを報告します。

拡張エラー・ファンクション・コール(59H)によって、エラー・コードの共通セットおよびエラー分類、位置、また推奨される対処方法などの、特定のエラー情報が得られます。ほとんどの重大な場合において、アプリケーションによりエラー・コードを分析し、特定の対処をすることができます。推奨される対処方法は、エラー・コードを理解できないプログラムを対照にしているものです。プログラムは、割り込み24Hの重大エラー・ハンドラーからか、または割り込み21Hのファンクション・コールを発行後、拡張エラー・サポートの利用で行うことができます。特定のエラー・コードに合わせて、コード化をしないようにしてください。

Control-Breakの対応

標準入出力の際にControl-Breakが起こると、割り込み23Hが発行されます。出力として∧C、復帰(CR)、および改行(LF)が生成される場合は、ファンクション・コール09Hおよび0AHが使用できます。

機能	機能番号
Control-Breakの動作の対応	23H
文字列の表示	09H
キーボード入力のバッファリング	0AH

標準入力、標準出力、標準プリンター、または非同期通信アダプターの動作の間Ctrl-Breakが入力されると、INT 23Hが実行されます。BREAKがオンであれば、06Hと07H以外のほとんどのファンクション・コールでINT 23Hがチェックされます。

ユーザーが作成したCtrl-Breakルーチンは∧C, 復帰(CR),および改行(LF)を出力として生成し、Control-Break動作に対しファンクション・コール09H, 0AHおよび0DHを使うことで対応します。ASCIIコード0DHと0AHは、それぞれ復帰(CR)と改行(LF)を表します。Ctrl-Breakルーチンが、すべてのレジスターを保存すると、IRET(割り込みからの復帰)命令で終了になり、プログラムの実行は継続になります。ルーチンがlongリターンで戻ってきたときは、キャリー・フラグを使って、実行を停止するかどうかを決めます。キャリー・フラグが設定されていないと、IRETの場合と同じように実行が継続になります。

IRETを使用しレジスターが変更にならなければ、Ctrl-Breakハンドラーの機能に対する制限は、特にありません。

システムの日付および時刻のリクエストと指定

システムの日付および時間に対して取得と設定を行うには、次の機能を使用します。

機能	機能番号
システムの日付のリクエスト	2AH
システムの日付の指定	2BH
システムの時刻のリクエスト	2CH
システムの時刻の指定	2DH

割り込みベクトルのリクエストおよび指定

プログラムは「割り込みベクトル」の内容を作成したり、変更することができます。割り込みベクトルは、ハードウエアとソフトウエアの割り込みをサービスする、メモリー上の４バイト・アドレスのルーチンです。出口ではプログラムは、最初に割り込みベクトルが指していたところに、リセットしなければなりません。

割り込みベクトルの内容を検査したり、内容を指定するプログラムが必要なときは、DOS/Vファンクション・コールの35Hと25Hを使用し、割り込みベクトルの場所の直接参照を、避けるようにしてください。

機能	機能番号
割り込みベクトルの値のリクエスト	35H
割り込みベクトルの値の指定	25H

第2部　プログラミング・ユーティリティーの使用

第7章 導入可能なディスク・ドライバーの作成

この章では、導入可能なディスク・ドライバーを作成するための指針と、システム・アーキテクチャー情報を説明します。

デバイス・ドライバーの種類

デバイス・ドライバーは、メモリー・イメージ・ファイルまたは装置機器として使用するために必要なコードを含む.EXEファイルです。DOS/Vでは、次の２種類のデバイス・ドライバーが導入できます。

- キャラクター・デバイス・ドライバー
- ブロック・デバイス・ドライバー

キャラクター・デバイス・ドライバー

キャラクター・デバイス・ドライバーは、シリアル方式で文字の入出力を行い、CON、AUX、CLOCK$のような名称です。キャラクター・装置へ入出力を行うために、多数のハンドルまたはFCBをオープンすることができます。なぜならキャラクター・デバイス・ドライバーは１つの名称しか持たないために、１つの装置しかサポートしないからです。

ブロック・デバイス・ドライバー

ブロック・デバイス・ドライバーは、固定ディスクまたはシステム上のディスケット・ドライブです。これらは、通常ディスクの物理セクター・サイズであるブロックと呼ばれる部分でランダム入出力を行います。ブロック装置は、キャラクター装置のように名前は付けられず、それらをオープンすることもできません。その代りに、A、B、Cのようなドライブ文字を使ってマップされます。

シングル・ブロック・デバイス・ドライバーは、複数のディスクまたはディスケット・ドライブに対応させることができます。たとえば、装置チェインの中で第一ブロック装置ドライバーは、A、B、C、Dのような４つのユニットを定義するかもしれません。第２のデバイス・ドライバーは、３つのユニットE、F、Gを定義するかもしれません。限界は26で、ドライブに割り当てられる文字AからZの装置です。チェインの中のドライバーの位置で、どのドライブ・ユニットとドライブ文字が一致するかが決定されます。

32MBを超えるメディアに対するサポート

DOS/V環境で作動するブロック・デバイス・ドライバーは、32ビット・セクター番号を処理するように書くことができます。この能力により32MBを超える固定ディスク・メディアがサポートされます。このサポートについては次に示す各節で記述します。

- 7-3ページの『デバイス・ドライバー・ヘッダー』
- 7-14ページの『BPB作成リクエスト』
- 7-17ページの『入出力リクエスト』

またDOS/Vはユーザー自身による、32MBを超えるメディアをサポートするデバイス・ドライバーの導入も可能です。

DOS/Vのデバイス・ドライバー導入方法

DOS/Vは、CONFIG.SYSのDEVICEコマンドを読み込み処理することで、始動時に動的にデバイス・ドライバーを導入します。たとえば、DRIVER1と呼ばれる装置ドライバーを書いた場合、CONFIG.SYSに次のコマンドを入れます。

```
device=driver1
```

DOS/V装置インターフェースは、チェイン中にデバイス・ドライバーを一緒にリンクするので、オプション装置にデバイス・ドライバーを加えることができます。

各装置は、初期化しなければなりません。装置ドライバーの初期化割り込みルーチンは、デバイス・ドライバーが導入されるときに一度コールされます。初期化ルーチンは、装置ドライバー導入の終了アドレスの、メモリーの位置を返します。終了アドレス・フィールドを設定した後、キャラクター装置ドライバーは状況ワードを設定し戻ります。

DOS/Vは、省略時装置を扱う前に導入されたキャラクター・デバイス・ドライバーを処理します。DOS/Vが新しいCON装置(たとえば、デバイス・ドライバー・ヘッダーの名称/ユニット・フィールド内に)を導入するためには、装置にCONと名づけ、属性フィールドに標準入力装置、および標準出力装置ビットを設定します。DOS/Vは、メモリーのどこにでもドライバーを導入しますから、セグメントではなく、位置の参照に注意しなければなりません。ドライバーは、毎回同じメモリー位置にロードされるとは限りません。

ブロック装置は、上記のキャラクター装置と同じ方法で導入されます。ブロック装置は、ユニット番号のような追加情報を返します。この番号が、装置の論理名を示します。たとえば、導入コールの時に現在の最大論理装置文字がFで、ブロック・デバイス・ドライバー初期化ルーチンが３つの論理ユニットを返す場合、装置の論理名はG、H、Iです。マッピングは、装置リストの内のドライバーの位置と、装置に関連したユニット番号によって決定されます。INITによって返されるユニット番号は、装置ヘッダーの名称/ユニット・フィールド中の値に上書きします。

ブロック装置はまたBIOSパラメーター・ブロック(BPB)配列にポインターを返します。これはｎワード・ポインターの配列へのポインターです。ここでｎは定義されたユニット番号です。すべてのユニットが同じならば、配列は同じBIOSパラメーター・ブロックを指すことができ、これによりスペースが節約できます。配列は、リターンによって設定される終了アドレス・ポインターの下で保護されなければなりません。

BPBはセクター・サイズおよび割り当てユニットごとのセクター数のような、装置に関する情報を含みます。BPBのセクター・サイズは、DOS/V初期化によって設定される最大許容サイズより、大きくなりません。

ブロック装置は、DOS/Vが特定のドライブ・ユニットに対して、どのパラメーターを使用しているかを報告するために、装置に渡されるメディア・ディスクリプター・バイトを返します。

デバイス・ドライバーの基本部分

デバイス・ドライバーは、装置を機器として使用するために必要なコードを含む、メモリー・イメージ・ファイルまたは.EXEファイルです。すべてのDOS/Vに導入可能な装置ドライバーは、共通して次のとおりです。

- デバイス・ドライバー・ヘッダー、これはDOS/Vに装置を指定し、ストラテジーおよび割り込みエントリー・ポイントを定義します。デバイス・ドライバーは単にロードされるだけで、プログラム・セグメント・プリフィックスを使用しないので、デバイス・ドライバー・ヘッダーはデバイス・ドライバーの物理的位置０(ORG０またはORGステートメント無し)にしなければなりません。
- ストラテジー・ルーチン、これは「リクエスト・ヘッダー」に対するポインターを保存します。
- 割り込みルーチン、これはリクエストされた処理を実行します。

デバイス・ドライバー・ヘッダー

デバイス・ドライバーは、以下を含むデバイス・ドライバー・ヘッダーを必要とします。

フィールド	**長さ**
次のデバイス・ドライバー・ヘッダーのポインター	DWORD
属性	WORD
ストラテジー・ルーチンへのポインター	WORD
割り込みルーチンへのポインター	WORD
名前/ユニット・フィールド	８ BYTES

次のデバイス・ドライバー・ヘッダーへのポインター

デバイス・ドライバー・ヘッダーは、次のデバイス・ドライバーのデバイス・ドライバー・ヘッダーへのポインターです。それは、デバイス・ドライバーがロードされる時にDOS/Vによって設定される、DWORDフィールドです。最初のワードはオフセットで、第２のワードはセグメントです。

単独のデバイス・ドライバーをロードしている場合、デバイス・ドライバーをロードする前にデバイス・ドライバー・ヘッダー・フィールドを-1に設定してください。複数のデバイス・ドライバーをロードしている場合は、デバイス・ドライバー・ヘッダー・フィールドの最初のワードを次のデバイス・ドライバーのヘッダーのオフセットに設定してください。最後のデバイス・ドライバーのデバイス・ドライバー・ヘッダー・フィールドには、-1を設定してください。

属性フィールド

属性フィールドは、デバイス・ドライバーの属性を指定します。

ビット15

ビット15は、装置がブロック装置かキャラクター装置かを指定します。ビット15が０に設定されている場合、ブロック装置を示します。ビット15が１に設定されている場合は、キャラクター・装置を示します。ビット15の設定は以後のビットの設定の解釈に影響することに注意してください。

ビット14

キャラクターおよびブロック装置の両方に対するビット14は、デバイス・ドライバーがIOCtl 44H、AL=2からAL=5によって、制御文字列(Control Strings)を扱えるかどうかをDOS/Vに伝えます。制御文字列が処理できる場合は、ビット14を１に設定してください。IOCtlサブ・ファンクションにより、デバイス・ドライバーは、それに渡される情報、たとえば、ボー・レートの設定またはフォームの長さの変更などを、標準の読み取りと書き込みを実行せずに解釈できます。制御文字列が処理されない時は、ビット14を０に設定してください。制御文字列を送受するためにIOCtlが発行され、ビット14が０に設定されているとDOS/Vはエラーを返します。

ビット13

ビット13は、ブロックおよびキャラクター装置の両方に使用されます。ブロック装置では、メディアがIBM形式の場合は、ビット13を０に設定してください。メディアが非IBM形式の場合は、ビット13を１に設定してください。キャラクター装置では、ドライバーが「Output-until-busy」をサポートしている場合、ビット13を０に設定してください。そうでない場合は、ビット13を１に設定してください。

「Output-until-busy」のサポートで、デバイス・ドライバーは、装置が準備されていれば、装置に文字を送ります。装置が準備されていない場合、デバイス・ドライバーはすぐにエラーを戻します。

ビット12

ビット12は予約されています。

ビット11

デバイス・ドライバー取り外し可能なメディアを扱う場合は、ビット11を設定してください。このビットは、オープン／クローズ取り外し可能メディア・ビットと呼ばれます。

ビット10から8

ビット10から８は予約されています。

ビット7

DOS/V以降のバージョンでは、デバイス・ドライバーIOCtlの問合せをサポートすることを示すためにこのビットを１に設定してください。このビットを１に設定すると、機能19H（すなわちIOCtlの問合せ）でドライバーが呼び出されます。

ビット6

ビット６は、キャラクターおよびブロック装置ドライバーの両方のための一般IOCtlビットです。このビットが１に設定されていると、デバイス・ドライバーは、一般IOCtlファンクション・コールをサポートします。このビットを１に設定することは、ブロック装置に「論理ドライブ」機能の取得／指定のサポートも示します。

ビット5と4

ビット５と４は予約されています。

ビット3

キャラクター装置がクロック装置の場合、ビット３を１に設定し、その他は０に設定してください。

ビット2

２キャラクター装置がNULL装置の場合、ビット２に１を設定し、その他は０を設定してください。ビットの設定は、NULL装置が使用されているかどうかをDOS/Vに知らせます。NULL装置は再割り当てできません。

ビット1

ビット15がキャラクター装置として１に設定されている場合、キャラクター装置が現行標準出力装置であることを示すために、ビット１を１に設定してください。ビット15がブロック装置として０に設定されている場合、32ビット・セクター番号のサポートを示すために、ビット１を１に設定してください。それ以外は、16ビット・セクター番号のサポートとみなされます。

ビット0

キャラクター装置が、現行標準入力装置である場合、ビット０を1に設定してください。現行標準入力装置でない場合は、ビット０を０に設定してください。

ストラテジー・ルーチンおよび割り込みルーチンへのポインター

DOS/Vがデバイス・ドライバーへリクエストを渡す時、それはデバイス・ドライバーを２度コールします。これら２つのフィールドは、第１および第２エントリー・ポイントを指します。それらはストラテジー・ルーチンと割り込みルーチンです。フィールドはワード値で、装置ヘッダーと同じセグメントになければなりません。

名前／ユニット・フィールド

これら８ビット・フィールドは、名前によりキャラクター装置を、またはユニットによりブロック装置を指定します。キャラクター装置名は、必要に応じて後ろに空白をつけて左寄せになります。ブロック装置ではDOS/Vが自動的に、INTコールによって返されるユニット番号の値でこのフィールドを埋めますが、最初の場所にユニット番号を置くことを選択してもかまいません。

ストラテジー・ルーチン

DOS/Vは、最初にストラテジー・ルーチンで装置ドライバーをコール、DOS/Vがデバイス・ドライバーに何をリクエストしているのかの情報を、リクエスト・パケットとして渡します。

ストラテジー・ルーチンはそのリクエストを実行しませんが、そのリクエストを蓄えるか、リクエスト・パケットのポインターを保存します。

割り込みルーチン

DOS/Vはストラテジー・ルーチンが返るとすぐに、パラメーター無しでデバイス・ドライバーの割り込みルーチンをコールします。割り込みルーチンの機能は、蓄えられたリクエストによって操作を行い、リクエスト・パケット中のデータを処理し、DOS/Vに返される情報を準備することです。

システム状態を保存することは、デバイス・ドライバー側の役目です。たとえばデバイス・ドライバーは、入力の時にすべてのレジスターを保管し、終了の時にそれらを復元しなければなりません。DOS/Vによって管理されているスタックが、全レジスターの保管に使用されます。さらにスタック空間が必要な場合は、追加スタックを割り当てそれを管理するのは、デバイス・ドライバーの役目です。

デバイス・ドライバーへのすべてのコールはFARコールです。FARのリターンはDOS/Vに戻るために、実行されなければなりません。

DOS/Vでのリクエストの渡し方

DOS/Vはリクエスト・パケットに、ES:BX中のポインターを渡します。そのパケットは、すべてのリクエストに共通な情報を含むリクエスト・ヘッダーと、それに続く、作成されているリクエストに関連するデータから構成されます。

リクエスト・ヘッダーの構造は、以下に示すとおりです。

フィールド	長さ
リクエスト・ヘッダーとそれに続くデータの長さ	BYTE
ブロック装置用のみのユニット・コード	BYTE

フィールド	長さ
コマンド・コード	BYTE
状況	WORD
予約済み	８-BYTE
データ	可変

長さのフィールド

長さのフィールドは、リクエスト・ヘッダーの長さと、その後のデータをバイトで指定します。

ユニット・コード・フィールド

ユニット・コード・フィールドは、ブロック装置ドライバーの中でリクエストしているユニットを指定します。ブロック・デバイス・ドライバーが定義された３つのユニットを持っている場合、たとえばユニット・コード・フィールドの可能な値は、０、１または２です。

コマンド・コード・フィールド

コマンド・コードはリクエストを指定します。コマンド・コード値とリクエストの記述の一覧は、7-8ページの『リクエストに対する応答』を参照してください。

状況フィールド

状況ワード・フィールドは、エントリーの時は０で、リターンの時にドライバー割り込みルーチンによって設定されます。

ビット15

ビット15はエラー・ビットです。ビット15が１に設定されると、状況ワードの下位８ビット(７～０)はエラー・コードを示します。

ビット14-10

ビット14から10は予約されています。

ビット9

ビット９は使用中(Busy)ビットです。状況リクエストコールへの応答として、キャラクター装置ドライバーは、装置が入出力リクエストを実行する準備ができているかどうかを示すために、使用中ビットを設定できます。ブロック装置ドライバーは、取り外し可能または不可能なメディアを示すために、使用中ビットを設定できます。コールについての詳細情報は、7-19ページの『キャラクター入出力状況リクエスト』と7-21ページの『取りはずし可能メディア・リクエスト』を参照してください。

ビット8

ビット８は、完了ビットです。設定されている場合は、操作が完了しています。ドライバーは、それが存在する時に完了ビットを１に設定します。

ビット7-0

ビット７から０は、状況ワードの下位８ビットです。ビット15が設定されている場合、ビット７から０はエラー・コードを含みます。エラー・コードとエラーの内容は次のとおりです。

コード	意味
00	書き込み保護違反
01	未定義のユニット
02	装置の準備ができていない
03	未定義のコマンド
04	CRCエラー
05	間違ったドライブ・リクエスト構造の長さ
06	シーク・エラー
07	未定義のメディア
08	セクターが見つからない
09	プリンター用紙切れ
0A	書き込み失敗
0B	読み取り失敗
0C	一般的失敗
0D	予約済み
0E	予約済み
0F	無効なディスク交換

リクエストに対する応答

デバイス・ドライバーに渡される各リクエスト・パケットは、どの機能を実行すべきかを伝えるためにリクエスト・ヘッダーに、コマンド・コード値を含みます。下に示す表は、DOS/V装置のインターフェース・コマンド・コード値と、これらの値がデータとともに渡されるときに、実行される機能を示しています。これらの機能の内いくつかは、ブロック装置またはキャラクター装置の、どちらかに対して特定されていることに注意してください。

この表に続いて、リクエスト・データ構造と割り込みルーチンが、何の実行を求められているかを詳細に説明します。これらの説明のいくつかは、複数のコマンド・コードに属します。

コマンド・コード	リクエストの説明	装置タイプ
0	初期化	両方
1	メディア・チェック	ブロック

コマンド・コード	リクエストの説明	装置タイプ
2	BPB作成	ブロック
3	IOCtl入力(属性のビット14が１に設定の場合のみコール)	両方
4	入力(読み込み)	両方
5	待ち時間無し非破壊入力	キャラクター
6	入力状況	キャラクター
7	入力フラッシュ	キャラクター
8	出力(書き込み)	両方
9	出力(ベリファイつき書き込み)	ブロック
10	出力状況	キャラクター
11	出力フラッシュ	キャラクター
12	IOCtl出力(属性のビット14が１に設定の場合のみコール)	両方
13	装置オープン(属性のビット11が１に設定の場合のみコール)	両方
14	装置クローズ(属性のビット11が１に設定の場合のみコール)	両方
15	取り外し可能メディア(属性のビット11が１に設定の場合のみコール)	ブロック
16	ビジーまでキャラクター出力	両方
19	一般IOCtlリクエスト(属性のビット６が１に設定の場合のみコール)	両方
23	論理装置取得(属性のビット６が１に設定の場合のみコール)	ブロック
24	論理装置設定(属性のビット６が１に設定の場合のみコール)	ブロック
25	IOCtlの問合せ（属性のビット７が１に設定の場合のみコール)	両方

初期化リクエスト

コマンド・コード = 0

フィールド	**長さ**
リクエスト・ヘッダー	13-BYTE
ユニットの数（キャラクター装置では設定されない)	BYTE
常駐プログラム・コードの終了アドレス	DWORD
BPB配列のポインター(キャラクター装置では設定されない)引数の剰余へのポインター	DWORD
ドライブ番号	BYTE
CONFIG.SYSエラー・メッセージ制御フラグ	WORD

ドライバーは次のことを行わなければなりません。

- ユニット数の設定(ブロック装置のみ)。
- BPB配列へのポインターの準備(ブロック装置のみ)。
- 初期化コードの実行(モデム、プリンター、その他へ)。
- 常駐プログラム・コードの終了アドレスの設定。
- リクエスト・ヘッダーへ状況ワードの設定。

初期化の際にCONFIG.SYSからデバイス・ドライバーへ渡される情報を得るために、BPBポインター・フィールドは、=に続いてCONFIG.SYSで渡される情報を含むバッファーを指します。この文字列は、復帰(0DH)または改行(0AH)のどちらかで終わります。この情報は読み取り専用です。システム・コール01Hから0CHと30Hがドライバーの初期化コードによって発行されます。

最後のバイト・パラメーターは、ブロック・ドライバーの最初のユニットのドライブ文字を含みます。たとえば、0=A、1=Bなどです。

初期化ルーチンが装置を設定できないことを決定し、メモリーを使用せずに終了したい場合、次の手順に従ってください。

- ユニット数を０に設定。
- 終了アドレス・オフセットを０に設定。
- コード・セグメントに終了アドレス・セグメントを設定。

DOS/Vは、デバイス・ドライバーをUMB（上位メモリー・ブロック）にロードするとき、ドライバーが利用できる最上位アドレスをINITリクエスト・パケットにセットします。この値は、デバイスのINIT機能が呼ばれる前に終了アドレス・フィールドにセットされます。その値は、ドライバーに対して割り振られているメモリー・ブロックの先頭のアドレスです。これはデバイスの初期化を完了する前に行われます。メモリー要

求は利用可能なスペースの総量に対してチェックされます。デバイスのコードとデータ要求のために十分なスペースがないときは、ロードは失敗します。

ブロック・デバイス・ドライバーは、サポートされるユニット単位に、ディスク・パラメーター・ブロックのために必要なスペースを調べなければなりません。ブロック・デバイス・ドライバーのために必要なスペースの総量は、次の式から求められます。

最終アドレス－（ユニット数×DPBサイズ）

複数のデバイス・ドライバーがひとつのメモリー・イメージ・ファイルにある場合、最後の初期化コールによって返された終了アドレスは、DOS/Vが使用しているアドレスと同じです。ひとつのメモリー・イメージ・ファイルの、すべての装置ドライバーが、同じ終了アドレスを返すようにすることを推奨します。

デバイス・ドライバーの初期化が失敗し、システムに**CONFIG.SYS の # 行目に誤りがあります**というエラー・メッセージを表示したい場合は、CONFIG.SYS内のエラー・メッセージ制御フラグを０以外の値に設定してください。

メディア・チェック・リクエスト

コマンド・コード = 1

フィールド	長さ
リクエスト・ヘッダー	13-BYTE
DOS/Vからのメディア・ディスクリプタ	BYTE
リターン	BYTE
前のボリュームIDのポインター(ビット11=1でディスク交換の場合に返される)	DWORD

コマンド・コード・フィールドが１の時、DOS/Vはドライブ装置にメディア・チェックをコールし、その現行メディア・ディスクリプタ・バイトを渡します。メディア・チェックは、次のうちの１つを返します。

- メディアが交換されていない
- メディアが交換された
- わからない
- エラー・コード

ドライバーは次のことを実行しなければなりません。

- リクエスト・ヘッダーに状況ワードを設定。
- リターン・バイトに次の値を設定。

-1　「メディアが交換された」場合

０　「わからない」な場合
１　「メディアが交換されていない」場合

ドライバーはリターン・バイトをどのように設定するかを決定するために、次の方法を使用します。

- メディアが取り外し不可能(固定ディスク)の場合、リターン・バイトに１を設定。
- 最後のアクセス成功から２秒に満たない場合、リターン・バイトに１を設定。
- チェンジラインが利用できない場合、リターン・バイトに０を設定します。
- チェンジラインが利用できるがアクティブでない場合、リターン・バイトに１を設定。
- 新しいBPBのメディア・バイトが、旧メディア・バイトに一致しない場合、リターン・バイトに-１を設定します。
- 現在のボリュームIDが以前のボリュームIDと一致する場合、またはシリアル番号が以前のシリアル番号と一致する場合、リターン・バイトに０を設定。

メディア・ディスクリプタ・バイト

現在はメディア・ディスクリプタ・バイトは、いくつかのメディア・タイプに対して定義されています。このバイトは、装置の非IBM形式ビットがオフの場合、メディア・バイトと同一でなければなりません。これらのあらかじめ定義された値は、次のとおりです。

```
メディア・ディスクリプタ
  バイト―>      1 1 1 1 1 x x x
  ビット―>      7 6 5 4 3 2 1 0
```

ビット	意味	
0	1=両面	0=両面でない
1	1=8セクター	0=8セクターでない
2	1=取り外し可能	0=取り外し不可能
3–7	１でなければなりません。	

注: 上述の例外は、メディア・タイプが定義されていないことを示すために使用されるF0Hと、両面で15セクター／トラックの5.25インチ・メディアのために使用される、F9Hのメディア・ディスクリプタ・バイト値です。

現行DOS/Vメディア・ディスクリプタ・バイトの例を示します。

ディスク・タイプ	#面	センター／トラック	メディア・ディスクイプタ
固定ディスク	--	--	F8H
5.25インチ	2	15	F9H
5.25インチ	1	9	FCH
5.25インチ	2	9	FDH
5.25インチ	1	8	FEH
5.25インチ	2	8	FFH
3.5インチ	2	9	F9H
3.5インチ	2	18	F0H
3.5インチ	2	36	F0H

片面または両面ディスケットのどちらを使用するかを決定するには、裏面を読み取りを試します。エラーが起これば、ディスケットは、片面と考えられます。メディア・ディスクリプタF0Hは、先に説明されなかったメディア・タイプに対して、使用されることがあります。プログラムは、メディアを区別するためにメディア・ディスクリプタを使用しないでください。DOS/V内部ルーチンは、IBM形式のディスケットのメディア・タイプを決定するために、BIOSパラメーター・ブロック(BPB)内の情報を使用します。これらのメディア・デイスクリプタ・バイトは、必ずしも固有のメディア・タイプを示しません。

BPB作成リクエスト

コマンド・コード = 2

フィールド	長さ
リクエスト・ヘッダー	13-BYTE
DOS/Vからのメディア・ディスクイプタ	BYTE
転送アドレス(バッファー・アドレス)	DWORD
BPBテーブルへのポインター	DWORD

DOS/Vは、次の２つの条件のもとでBPB(BIOSパラメーター・ブロック)作成をコールします。

- 「メディアが交換された」が返される場合。
- 「わからない」が返され、バッファーが使用されていない場合。使用されたバッファーは、ディスクにまだ書き込まれていない、変更したデータを含むバッファーです。

ドライバーは次のことを実行しなければなりません。

- BPBにポインターを設定。
- リクエスト・ヘッダーに状況ワードを設定。

ドライバーは、BPBテーブルへポインターを返すために、ユニットの中にあるメディア・タイプを決定しなければなりません。FAT IDバイトは、メディアの構造とレイアウトを決定しました。FAT IDバイトは、わずか８つの値(F8HからFFH)しか収容できず、新しいメディア・タイプが作られると、利用できる値はすぐなくなることが予想されます。さまざまなメディア・レイアウトがある場合、FATやディレクトリーを読み込むことをリクエストする前にDOS/VはFATの位置とディレクトリーを知る必要があります。

次の段落で、DOS/Vがメディア・タイプを決定する方法を説明します。

特定のメディアのBPBに関する情報は、メディアのブート・セクターに保持されています。ブート・セクターの形式の要約は、次のとおりです。

フィールド	長さ
後にNOP命令(90H)の続く２バイトのshort JMP命令(EBH)	３ BYTES
製品名とバージョン	８ BYTES
セクター当りのバイト、２の累乗でなければなりません。	WORD

フィールド	長さ
割り振りユニット当りのセクター、２の累乗でなければなりません。	BYTE
論理セクター０で始まる予約セクター	WORD
FAT数	BYTE
ルート・ディレクトリー・エントリーの最大数	WORD
ブート・セクター、FAT域、ディレクトリーを含むメディアの全セクター数	WORD
メディア・ディスクリプタ	BYTE
FATが占めているセクター数	WORD
トラック当りのセクター数	WORD
ヘッド数	WORD
隠しセクター数	DWORD

ブート・セクターに含まれるBPB情報は、「セクター当りのバイト」エントリーで始まります。最後の３ワードは、デバイス・ドライバーがメディアを識別するのを助けるためのものです。ヘッド数は記憶容量は同じで、サーフェース数が異なる、別の複数ヘッド・ドライブをサポートするのに便利です。隠しセクター数は、ドライブ区画スキームをサポートするのに便利です。

ボリュームIDとディスク交換をサポートするドライバーに対して、このコールで新しいボリュームIDをディスクから読み込めます。このコールは、ディスクが容認された方法で交換されたことを示します。

32MBを超える区画、または32MB境界を超えて開始する区画、32MB境界を横切る区画を扱うために、DOS/Vは拡張BPB構造を定義します。メディアのサイズによって、既存のBPBまたはセクターの区画サイズを示すための、追加DWORDフィールドを含む拡張BPBのいずれかを使用できます。

ブロック・デバイス・ドライバー・ヘッダーの属性フィールドのビット１は、装置が32ビット・セクター数を処理できるかどうかを示します。32ビット・サポートを示すには、ビット１を設定して下さい。

フィールド	長さ
セクター当りのバイト数	WORD
割り当てユニット当りのセクター	BYTE

フィールド	長さ
論理セクター０で始まる予約セクター	WORD
FAT数、FATシステムがなければ０	BYTE
ルート、ディレクトリー・エントリーの最大数	WORD
メディアの全セクター数。このフィールドは32MB未満の区画を定義するために使用されます。このフィールドに０を設定すると、下に示すメディアの全セクター数(32ビット)を使用することを示します。	WORD
メディア・ディスクリプタ	BYTE
FATが占めているセクター数	WORD
トラック当りのセクター数	WORD
ヘッド数	WORD
隠しセクター数	DWORD
メディアの全セクター数(32ビット)。このフィールドは、32MBを超える、または32MB境界を横切る区画を定義するために使用されます。	DWORD

拡張BPBは従来のBPB構造のスーパーセットです。この構造と従来のBPBの構造の互換性を最大限に活用するために、DOS/Vは次の規則を使用します。

- メディアの全セクター数に、隠しセクター数を加えて64KBを超える場合、メディア・エントリー(DWORD)の32ビット全セクター数(DWORD)を使用します。
- その他には、**メディア・エントリーの全セクター数** (WORD)を使用します。

ブート・レコードは、各ディスク区画と各拡張DOS/V区画ボリュームの初めに存在します。DOS/Vは、自動的に拡張ブート・レコードを作成します。拡張ブート・レコードの形式は次のとおりです。

フィールド	長さ
後に１つのNOP命令(90H)の続く２バイトのshort JMP命令(EBH)	３ BYTES
製品名とバージョン	８ BYTES
拡張BPB	25 BYTES
物理ドライブ番号	BYTE

フィールド	長さ
予約済み	BYTE
拡張ブート・レコード・シグネチャー	BYTE
ボリューム・シリアル番号	DWORD
ボリューム・ラベル	11 BYTES
予約済み	８ BYTES

注: 拡張ブート・レコード・シグネチャーの値は29Hです。物理ドライブ番号の値は、常に0Hまたは80Hです。

リクエスト・ヘッダーにセクター番号を持つ、拡張ドライバーへの全リクエストの場合、セクター番号はDWORDです。標準DOS/Vブロック・デバイス・ドライバーは、32ビット・サポートのために、属性のビット１を設定します。

デバイス・ドライバーへの各コールについて、DOS/Vはリクエスト中で渡される開始セクター番号が、デバイス・ドライバーでサポートされているかどうかを見るためにチェックします。この値が、旧スタイルの装置ドライバーに対して64KBを越えている場合、DOS/Vは「未定義のメディア(Unknown media: 07H)」のデバイス・ドライバー・エラーを返します。

入出力リクエスト

コマンド・コード = ３、４、８、９、１２

フィールド	長さ
リクエスト・ヘッダー	13-BYTE
メディア・ディスクリプター・バイト	BYTE
転送アドレス(バッファー・アドレス)	DWORD
バイト／セクター・カウント	WORD
開始セクター番号(１の場合、DWORD開始セクター番号を使用します。このエントリーは、キャラクター装置に対しては意味がありません。)	WORD
DOS 3.0からDOS/Vに対するエラーコード0FHが返される場合の、ボリュームIDのポインター	DWORD

フィールド	**長さ**
開始32ビットセクター番号（属性のビット１が設定されているブロック・デバイス・ドライバーにこのエントリーを使用します。）	DWORD

DOS/V**入出力リクエスト**構造は、32ビット・セクター番号を処理でき、４億セクター以上のメディアのサポートをします。

ドライバーは次のことを行わなければなりません。

- リクエスト・ヘッダーの状況ワードの設定
- リクエストされた機能の実行
- 転送される実際のセクターまたはバイト数の設定

エラー・チェックはIOCtlコールでは行われません。しかしデバイス・ドライバーは、転送される実際のバイト数にセクターまたはバイト・カウントを設定しなければなりません。

状況によっては、ブロック・デバイス・ドライバーは、装置ドライバー・リクエスト・パケット内で転送アドレスの**循環(** *wrap around*)があると思われる64KBのWRITE操作の実行がリクエストされるかもしれません。これはDOS/VのWRITEコードに最適化が追加されるために起こります。それは、ファイルの現在の終端を超えて拡張されているファイルで、64KBセクター・サイズ以内のWRITEの際のみに起こります。ブロック・デバイス・ドライバーは、循環するWRITEのバランスを無視することを許されています。たとえば、XXXX:1の転送アドレスを持つセクターの10000HバイトのWRITEでは、最後の２バイトを無視します。

DOS/Vファクション・コールを使用するプログラムは、転送バッファー・セグメントで**循環**が行われなければならないために、FFFFHバイト以上の入出力操作をリクエストできないことに気をつけてください。転送セグメントで**循環**したバイトは、無視できます。

ドライバーが0FH「無効なディスク交換(Invalid Disk Change)」のエラー・コードを返す場合、それは正しいボリュームIDを識別するASCIIZ文字列への、DWORDポインターを与えなければなりません。システムは、ディスクを再び差し込むように指示します。

OPENおよびCLOSEコールによって管理される、ディスク上のオープン・ファイルの参照用カウントにより、ドライバーにいつエラー0FHを返すかを決定させます。開いているファイルがなくて(参照用カウント=０)ディスクが交換された場合、I/Oは有効でありエラー0FHは返されません。オープン・ファイル(参照用カウント>０)があり、ディスクが交換された場合、エラー0FHの状態が存在することがあります。DOS/V IBMDOS.COMまたはMSDOS.SYSは、SHAREがロードされている場合にのみ、OPENまたはCLOSE機能をリクエストします。

待ち時間無し非破壊入力リクエスト

コマンド・コード = 5

フィールド	長さ
リクエスト・ヘッダー	13-BYTE
装置から読み込んだバイト	BYTE

ドライバーは次のことを行わなければなりません。

- 装置から１バイトを返す
- リクエスト・ヘッダーに状況ワードを設定

キャラクター装置がバッファーに文字のあることを意味する、使用中ビット=０を返す場合、読み込まれる次の文字が返されます。この文字は、入力バッファーから消去されません(つまり、非破壊入力)。このコールで、DOS/Vは１つの先行入力を受けつけます。

キャラクター入出力状況リクエスト

コマンド・コード = ６、１０

フィールド	長さ
リクエスト・ヘッダー	13-BYTE

ドライバーは次のことを行わなければなりません。

- リクエストされた機能の実行
- 使用中ビットの設定
- リクエスト・ヘッダーに状況ワードを設定

使用中ビットは、出力と入力に対して異なって設定されます。

キャラクター装置への出力

リターンの際に使用中ビットが１の場合、書き込みリクエストは現在のリクエストの完了を待ちます。使用中ビットが０の場合、リクエストは待たず実行もしません。したがって、書き込みリクエストはすぐに開始します。

バッファ付きキャラクター装置の入力

リターンの際に使用中ビットが１の場合、読み取りリクエストは物理装置へ行きます。使用中ビットが０の場合、キャラクターは装置バッファーにあり、読み取りはすぐに返ります。これは、ユーザーが何かをタイプしたことも示します。DOS/Vはすべてのキャラクター装置が、先行タイプ入力バッファーを持っているとみなします。このバッファーを持たない装置は、DOS/Vが存在しないバッファーに入れる情報を待ちながら、永久にループしないように、使用中ビット=０を常に返してください。

キャラクター入出力フラッシュ・リクエスト

コマンド・コード=７、１１

フィールド	長さ
リクエスト・ヘッダー	13-BYTE

このコールはドライバーが関知している、すべての未解決リクエストをフラッシュ(終了)することを知らせます。その主な使用は、キャラクター装置の入力待ち行列をフラッシュすることです。

ドライバーは、リターンの際にリクエスト・ヘッダーに状況ワードを設定しなければなりません。

オープンとクローズ・リクエスト

コマンド・コード = １３、１４

フィールド	長さ
リクエスト・ヘッダー	13-BYTE

これらのコールは属性ワードのビット11が設定されている場合、装置上での現行ファイル活動状態についての情報を、装置に伝えるためのものです。

ブロック装置では、これらのコールはローカル・バッファリングを管理するために使用できます。装置は、参照用カウントを保持できます。オープンするたびに参照用カウントが増えます。クローズするたびに参照用カウントが減ります。参照用カウントが０の時、装置上でオープン・ファイルはありません。そのため装置は、それが書き込まれた装置内のバッファーをフラッシュしてください。なぜならメディアを、取り外し可能なメディア・ドライブ上で交換できるからです。メディアが交換された場合、バッファーをフラッシュせずに参照用カウントを０にリセットしてください。

これらのコールは、キャラクター装置ではさらに便利です。OPENコールは、初期化文字列を装置に送ることができます。プリンターでは、コールは、フォントまたはページサイズを設定するための文字列を送ることができ、これによりプリンターはI/Oストリームの開始のところで、常に既知状態になります。

同様にCLOSEコールは、I/Oストリームの終わりにポスト文字列、たとえば「用紙送り」を送ることができます。

準備および終了文字列を設定するためにIOCtlを使用することは、シリアルI/O装置ストリーム制御に柔軟性に富んだメカニズムを提供します。

取りはずし可能メディア・リクエスト

コマンド・コード = １５

フィールド	長さ
リクエスト・ヘッダー	13-BYTE

このコールを使用するために、属性フィールドのビット11を１に設定してください。ブロック装置は、IOCtlファンクション・コール(44H)のサブファンクションによってのみ、この機能を使用することができます。

このコールは、取り外し可能と取り外し不可能なメディア・ドライブのどちらを扱っているかを、ユーティリティーに伝えるのに便利です。たとえばFORMATユーティリティーは、いくつかのプロンプトの異なるバージョンを表示しますので、ドライブが取り外し可能か、不可能かを知る必要があります。

情報は状況ワードの使用中ビットで返されます。使用中ビットが１の場合、メディアは取り外し不可能です。使用中ビットが０の場合、メディアは取り外し可能です。

エラー・ビットのチェックは行われません。このコールは、常に成功するとみなされます。

ビジーまで出力

コマンド・コード = １６

フィールド	長さ
リクエスト・ヘッダー	13-BYTE
転送アドレス	DWORD
バイト・カウント	WORD

ドライバーは次のことを行わなければなりません。

- リクエスト・ヘッダーに状況ワードを設定
- バイト・カウント・ワードに転送される実際のバイト数

この機能は、デバイスがビジーになるまで、指定されたメモリー・バッファーからデバイスにデータを転送します。この機能が呼び出されるのは、デバイス・ヘッダーのなかのデバイス属性ワードのビット13がセットされている場合だけです。この機能を使って実際に転送したバイト数が要求されたバイト数よりも少なくてもエラーではありません。

一般IOCtlリクエスト

コマンド・コード = １９

フィールド	長さ
リクエスト・ヘッダー	13-BYTE
メジャー機能	BYTE
マイナー機能	BYTE
SIの内容	WORD
DIの内容	WORD
一般IOCtlリクエスト・パケットのポインター	DWORD

ドライバーは次のことを行わなければなりません。

- 一般IOCtlリクエストのもとで記述された機能のサポート
- それ自身のトラック表(TrackLayout)の保守

一般IOCtlリクエストによって提供される機能の説明については、C-1ページの付録C、『装置の入出力制御(IOCtl)』を参照してください。

論理装置の取得リクエスト

コマンド・コード = ２３

フィールド	長さ
リクエスト・ヘッダー長（下の注を参照）	BYTE
入力(ユニット・コード)	BYTE
コマンド・コード	BYTE
状況	WORD
予約済み	8 BYTE

この機能から戻ったとき、ユニット・コードは最後に参照したユニットかまたはゼロになっています。また、状況ワードは更新されています。

論理装置の設定リクエスト

コマンド・コード = 2 4

フィールド	長さ
リクエスト・ヘッダー長（下の注を参照）	BYTE
入力(ユニット・コード)	BYTE
コマンド・コード	BYTE
状況	WORD
予約済み	8 BYTE

この機能から戻ったとき、状況ワードは更新されています。

注: 要求ヘッダー長の値は、長さフィールドそのもののバイト数も含んでいます。

IOCtlの問合せ

コマンド・コード = 2 5

フィールド	長さ
リクエスト・ヘッダー	13-BYTE
メジャー機能	BYTE
マイナー機能	BYTE
SIの内容	WORD
DIの内容	WORD
一般IOCtlリクエスト・パケットのポインター	DWORD

ドライバーは次のことを行わなければなりません。

- 一般IOCtlリクエストのもとで記述された機能のサポート
- それ自身のトラック表(TrackLayout)の保守

ドライバーは、デバイス属性ワードのビット７をセットすることによって、IOCtlの問合せをサポートすることを明示します。このビットがセットされると、機能19Hと一般IOCtlリクエスト・パケットによってドライバーは呼び出されます。もしセットされないと、ドライバーはIOCtlの問合せ呼出しを受け取ることはありません。

ドライバーは、リクエスト・パケットのなかのカテゴリー・コードと機能番号をチェックし、呼出しをハンドルできるときはエラーなしのシグナルを返さなければなりませ

ん。呼出しをハンドルできないときは、ドライバーは「未知のコマンドである（エラー・コード３）」というエラー・コードを返さなければなりません。通常は、一般IOCtl呼び出しを使用するプログラムはDOS 3.2での使用に加えて、最初にIOCtl Handleの問合せまたはIOCtl Deviceの問合せを呼び出します。そのあと、特定の呼出しがサポートされているかどうかが判別され、最終的には実際の機能が呼び出されます。

一般IOCtlリクエストによって提供される機能の説明については、C-1ページの付録C、『装置の入出力制御(IOCtl)』を参照してください。

CLOCK$デバイス・ドライバーの例

この機能は「リアル・タイム時計」ボードのことです。このボードをTIMEおよびDATEシステムと統合するように、特殊デバイス・ドライバーとして、属性ワードをCLOCK$装置と指定します。このデバイス・ドライバーは、他のキャラクター・デバイス・ドライバーのように機能を定義し実行します。多くの機能は完了ビット、リセット・エラー・ビット、リターンで設定されます。この装置に読み取りまたは書き込みが起こると、６バイトが転送されます。最初の２バイトは、80年１月１日からの日数を示すワードです。３バイト目は分、４バイト目は時間、５バイト目は１/100秒、６バイト目は秒です。CLOCK$装置を読むと日付と時刻を取得でき、それに書き込むことによって日付と時刻を設定できます。

第3部　付録

付録A.　DOS/V割り込み

ここでは、DOS/V割り込みの使い方について説明します。

割り込みのタイプとして20Hから3FHまでがDOS/Vにより予約されています。また絶対メモリー番地80HからFFHまでが、DOS/Vによって予約されています。なおすべての割り込み値は16進数です。

20Hプログラムの終了

プログラムを終了するには割り込み20Hを発行します。このベクトルは、終了アドレス、Ctrl-Breakアドレス、重大エラー出口アドレスの値をプログラムのエントリー時にもっていた値に復元するためにDOS/V内のロジックに転送します。すべてのファイル・バッファーは、フラッシュされ、すべてのハンドルはクローズされます。この割り込みを発行する前に長さを変えられたすべてのファイル(ファンクション・コール10Hと3EHを参照)をクローズしなければなりません。変更されたファイルをクローズしないとその長さ、日付、時刻は、ディレクトリーに正確に記録されません。

終了の時に完了コードまたはエラー・コードを渡すプログラムでは、ファンクション・コール4CH(プロセスの終了)または31H(常駐のままプロセス終了)のいずれかを使用しなければなりません。これら２つの方法は、割り込み20Hを使用するより望ましい方法です。またこれら２つの方法によって返されるコードは、バッチ処理で分岐条件に用いられます。バッチ処理のERRORLEVELサブコマンドに関する情報についてはファンクション・コール4CHを参照してください。

重要:割り込み20Hを発生する前に、プログラムはCSレジスターはそのプログラム・セグメント・プリフィックスのアドレスを含まなければなりません。

21Hファンクション・リクエスト

B-1ページの付録B、『DOS/Vファンクション・コール』の21H内で発行される各ファクション・コールを参照してください。

22H終了アドレス

プログラムが終了するとき、この割り込みでアドレスに制御が移されます。このアドレスは、セグメントが作成される時にプログラムのプログラム・セグメント・プリフィックスに複写されます。EXECファンクション・コールがこれを行いますので、ユーザーはこの割り込みを発行しないようにしてください。

23H Ctrl-Break出口アドレス

もし、ユーザーが標準入力、標準出力、標準プリンターまたは非同期通信アダプター操作の間にCtrlキーとBreakキーを押すと、割り込み23Hが実行されます。BREAKがONなら、割り込み23H(06Hと07Hのコールを除いて)は、ほとんどのファンクション・コールに対してチェックされます。ユーザーの書いたCtrl-Breakルーチンがすべてのレジスターを保存するなら、プログラム実行を継続するために、割り込みからのリターン命令(IRET)で終わることが可能です。

ユーザーの書いた割り込みプログラムがlong returnで返る場合、キャリー・フラグがプログラムを終了するか否かを決定するために使用されます。キャリー・フラグ桁上げが設定されるとプログラムは終了し、そうでない場合は(IRETによるリターンのように)実行は継続します。

ユーザーの書いたCtrl-Break割り込みが、ファンクション・コール09Hまたは0AHを使用すると、∧C、復帰、行送りが出力されます。IRETで実行が継続される場合、I/Oはその行の最初から続けられます。

割り込みが起きるとすべてのレジスターは、DOS/V元のファンクション・コールのときに持っていた値に設定されます。IRETが使用される場合、レジスターが変更されない限り、DOS/Vファンクション・コールも含めて、Ctrl-Breakハンドラーができることの制限はありません。

プログラムが新しいセグメントを作成し、次のプログラム内にロードするとき、Ctrl-Breakアドレスを変更します。２番目のプログラムが終了し最初のプログラムに戻ると、Ctrl-Breakアドレスは第２のプログラムの実行前の値に復元されます。それは、２番目のプログラムのプログラム・セグメント・プリフィックスから復元されます。この割り込みは発行しないようにしてください。

24H重大エラー・ハンドラー

重大エラーは、DOSのファンクション・コールが実行できない時に返されます。このエラーは、ハードウェアの状態、たとえばプリンターの用紙切れ、ディスケット・ドライブのドアが開いている、またはディスケットの空き空間不足などによって起こります。DOS/Vの中で重大エラーが起こると制御は割り込み24Hに移されます。エラー・ハンドラーのエントリーで、エラーがディスク・エラーの場合(もっとも頻出度が高い)、AHのビット7=0 (最上位ビット)です。そうでなければ、ビット7=1です。

BP:SIは、追加情報を取り出せる装置ヘッダー制御ブロックのアドレスが含まれます。A-6を参照してください。

レジスターは、再試行操作に対して設定され、エラー・コードは上位半分が未定義であるDIレジスターの下位半分に入ります。エラー・コードは、次のとおりです。

エラー・コード	意味
0	書き込み禁止のディスケットに書き込もうとした
1	未定義ユニット
2	ドライブの準備ができていない
3	未定義コマンド
4	データ・エラー(CRC)
5	間違ったリクエスト構造長
6	シーク・エラー
7	未定義のメディア・タイプ
8	セレクターが見つからない
9	プリンターの用紙切れ
A	書き込み失敗
B	読み込み失敗
C	一般エラー

ユーザー・スタックは有効で上部から下部まで次の内容です。

IP	INT24H発行によるDOS/Vレジスター
CS	
FLAGS	
AX	元のINT21Hリクエスト時のユーザー・レジスター
BX	
CX	
DX	
SI	
DI	
BP	
DS	
ES	
IP	INT 21Hを発行した次のアドレス（リターン・アドレス）
CS	
FLAGS	

レジスターは、たとえばIRETが実行される場合に設定され、その場合DOS/Vは次のように(AL)によって応答します。

(AL) ＝0 エラーを無視
＝1 操作を再試行
＝2 割り込み22Hを通してプログラムを終了
＝3 実行中にシステム呼び出し失敗

注: 「エラーを無視」の応答を選択するときは注意してください。これによりDOS/Vは完全に処理を完了していなくても、完了したと信じてしまいます。

重大エラー・ハンドラーからユーザー・エラー・ルーチンに制御を戻すには、次のことをしなければなりません。

INT24Hが起きる前に:

1. ユーザー・アプリケーション・プログラム初期化コードは、INT24Hベクトルを保存し、ユーザー・エラー・ルーチンを指すベクトルでそれを置き換えます。

INT24Hが起きたとき:

2. ユーザー・エラー・ルーチンが制御を受け取ると、スタックにフラグ・レジスターをプッシュさせ、ステップ１で保存したオリジナルのINT24HベクトルへのCALL FARを実行します。

3. DOS/Vは適当なプロンプトを与え、ユーザーの入力(中止、再試行、失敗、または強行)を待ちます。ユーザーの入力の後、DOS/Vは、CALL FARに続く命令でユーザー・エラー・ルーチンに制御を戻します。

4. ユーザー・エラー・ルーチンは、必要とするどんなタスクでも発行することができます。エラーの発生した地点でオリジナルのアプリケーション・プログラムに戻るには、エラー・ルーチンでIRET命令を実行する必要があります。そうでないと、ユーザー・エラー・ルーチンは、IP, CS, フラグ・レジスターをスタックから除去しなければなりません。その後、制御は望ましい場所に渡されます。

ディスク・エラー

ディスクにハード・エラー(AHのビット７=０)があれば、レジスターALは、障害のあったドライブ番号(０=ドライブAなど)を含みます。AHのビット０～２は、影響を受けたディスクの域およびそれが読み取りまたは書き込み操作のいずれであるかを次のように示します。

ビット０ =0 読み取り操作の場合
=1 書き込み操作の場合

Bits ２～１(影響を受けたディスク領域)

０ 0 DOS/V領域
０ 1 FAT(ファイル割り当てテーブル)
１ 0 ディレクトリー
１ 1 データ域

AHビット３～５でどの応答が有効かを示します。それは以下のとおりです。

ビット３ =0 「失敗」が許されない場合
=1 「失敗」が許される場合
ビット４ =0 「再試行」が許されない場合
=1 「再試行」が許される場合
ビット５ =0 「エラーを無視」が許可されない場合
=1 「エラーを無視」が許される場合

無効な応答の取り扱い

「エラーを無視」が指定 (AL=０)され、「エラーを無視」が許されない(ビット５=０)場合には、「失敗」 (AL=３)を応答します。

「再試行」が指定(AL=１)され、「再試行」が許されない(ビット４=０)場合には、「失敗」(AL=３)を応答します。

「失敗」が指定(AL=３)され、「失敗」が容認されない(ビット３=０)場合には、「中止」(AL=２)を応答します。

他のエラー

AHのビット７=１の場合、エラーがキャラクター装置に起こったか、またはFATの間違ったメモリー・イメージの結果です。BP:SIで渡されるデバイス・ドライバー・ヘッダーを調べれば、どちらの場合であるかが分かります。属性バイトの高位ビットがブロック装置を示している場合、エラーは、不良FATによるものです。そうでなければ、エラーはキャラクター装置によるものです。

キャラクター装置の場合は、ALレジスターの内容は予測できません。エラー・コードは上述のようにDIの中にあります。

注:

1. このルーチンの制御をディスク・エラーに渡す前に、５回再試行してください。エラーがFATまたはディレクトリーにある時は３回再試行してください。
2. ディスク・エラーに対して、この終了は割り込み21Hのファンクション・コールの時に起こるエラーに対してのみ行われます。割り込み25Hまたは26Hの時のエラーには使用されません。
3. このルーチンは、割り込み禁止状態で入力されます。
4. すべてのレジスターを、保存しなければなりません。
5. この割り込みハンドラーでは、DOS/Vのファンクション・コールを使用しないようにすべきです。必要ならば、01Hから0CH, 30H, 59Hのコールを使用してください。他のいずれのコールもDOS/Vのスタックを破壊し、DOS/Vを予測不可能な状態にします。
6. 割り込みハンドラーが、デバイス・ドライバー・ヘッダーの内容を変更しないようにしてください。
7. 割り込みハンドラーがDOS/Vに戻らず、それ自身でエラーを扱う場合、スタックからアプリケーション・プログラムのレジスターを復元し、スタック上の最後の３単語を除くすべてを取り除き、それからIRETを発行しなければなりません。これは、エラーの起こったINT21H直後のプログラムに戻ります。これが行われた場合には、DOS/Vは0CHより高位のファンクション・コールが発行されるまで不安定な状態になるのでご注意してください。そのためお勧めできません。

重大エラーを終了する場合の適切な方法は、「失敗」を使用し、それからINT 21Hの拡張エラー・コードをテストすることです。

8. 「エラーを無視」リクエスト(AL=０)は、FATまたはDIRセクターに起こる重大エラーは「失敗」に変換されます。

9. さらにエラー情報を得る方法についての詳細は、B-6ページの『エラーに対する応答』とB-6ページの『拡張エラー・コード』を参照してください。

10. DOS/Vでは、「エラーを無視」リクエスト(AL=０)は、ある重大エラー(50〜79)に対しては「失敗」リクエストに変換されます。

BP:SIによって示されるデバイス・ドライバー・ヘッダーは、次のような形式です。

次の装置へのDWORDポインター(最後の装置ならFFFFH)
WORD属性 ビット15 = １ ならばキャラクター装置 = ０ ならばブロック装置 ビットが15が１の場合 ビット０ = １ ならば現行標準入力 ビット１ = １ ならば現行標準出力 ビット２ = １ ならば現行NULL装置 ビット３ = １ ならば現行CLOCK装置 ビット14 IOCtlビットです。
デバイス・ドライバー・ストラテジー・エントリー・ポイントへのWORDポインター
デバイス・ドライバー割り込みエントリー・ポイントへのWORDポインター
ブロック装置に対するフィールドを指定された８-BYTEキャラクター装置最初のバイトは、ユニットの数

エラーがブロック装置またはキャラクター装置に発生したかどうかを調べるには属性フィールド(BP:SI＋４のWORD)のビット15を見てください。

キャラクター装置の名称は、BP:SI＋10から始まる８バイトを見れば分かります。

25H/26H絶対ディスクの読み取り／書き込み

割り込みベクトル25Hと26Hは、デバイス・ドライバーに制御を移します。これらは、32MB以上のサイズのメディアを直接アクセスできるように拡張されました。これらによるCXレジスタの使用で、規定の25Hおよび26H割込と区別します。規定の形式パラ

メーターが32MB以上のメディアにアクセスしようとする時に使用された場合、0207HのエラーコードがAXで返されます。

拡張された25Hまたは26Hに対するリクエストは、次のようにします。

```
        MOV     AL,DRIVE          ; プロセスするドライバー番号
                                  ; 0 = A
                                  ; 1 = B
                                  ; 2 = C . . .
        MOV     BX,SEG PACKET     ; パラメーター・リスト
        MOV     DS,BX
        MOV     BX,OFFSET PACKET
        MOV     CX,-1             ; 拡張形式の指定
        INT     25H or 26H        ; DOS/Vへリクエストの発行
        POP     AX                ; スタック・ワードの削除
        JC      ERROR             ; AXで返されるリターン・コード

PACKET  LABEL   BYTE              ; 制御パケット
        DD      RBA               ; 最初のセクターのRBA
                                  ; (0 origin)
        DW      COUNT             ; 入出力するセクター数
        DD      BUFFER            ; データ・バッファ
```

リターン時、元のフラグはまだスタックにあります(INT命令によってそこに退避されたままです)。この元のフラグは、リターン情報が現行フラグで渡されるので必要です。スタックが制御不能でむやみに大きくならないように必ずスタックをポップしてください。

警告: この割り込みによって扱われるディスクI/Oが64KBのダイレクト・メモリー・アクセスによって限界を超える場合、予測できない結果が起こる可能性があります。コールを発行する前に読み書きの行われるセクター・サイズとセクター番号を注意してチェックすることをお勧めします。

指定されるセクター数が、特定のドライブから転送アドレスに転送されます。論理セクター番号(LSN)は、トラック０、ヘッド０、セクター１(論理セクター０)から始め、同じヘッドに沿って連続的に各セクターを数え、それから次のヘッドへと数え最後のヘッドの最後のセクターまで数えることによって得られます。したがって、論理セクター１はトラック０、ヘッド０、セクター２で、論理セクター２はトラック０、ヘッド０、セクター３というようになります。続いて、次のトラックのヘッド０のセクター１から数え続けます。セクターは連続的に数えられますが、(たとえば、上述の例のトラック０のセクター２とセクター３)、それらはインターリービングのために、ディスク上で物理的に隣合っている必要のないことに注意してください。マッピングは、DOSバージョン1.10で使用される両面ディスケットとは異なることにも注意してください。。

セグメント・レジスターを除くすべてのレジスターは、このコールによって破壊されます。転送が成功すればキャリー・フラグ(CF)は０です。もし失敗すればCF=１となり、(AX)は次のようにエラーを示します。(AL)は、DOS/Vのエラーコードで、これは

割り込み24Hが発行される時のDIの下位バイトで返されるものと同じです。(AH)は次のコードを含みます。

80H 接続機構が応答障害
40H SEEK操作障害
08H ディスケット読み込み時の不良CRC
04H リクエストされたセクターが見つからない
03H 書き込み禁止ディスケットに書き込もうとした
02H 上に示された以外のエラー

警告: 取り外し可能なメディアにこの割り込みを発行する前に、ドライブの中のメディアが正しくセットされていなければなりません。これは、BUILD BPBリターンでBPBを返すリクエストをするINT21H一般IOCTL (AH=44H)、またはINIT21H現行ディレクトリーの取得(AH=47H)、のいずれかを発行することによって実現できます。

27H常駐のまま終了

このベクトルは、COMMAND.COMが制御を復帰する時に常駐のまま残るプログラムによって使用されます。

DOS/Vファンクション・コール31Hは、プログラムを常駐のまま残すために推奨される方法です。なぜならこれはリターン情報を渡せるからです。また64KB以上のプログラムを常駐のままに残すことができます。自身を初期設定した後、プログラムはDXにそのプログラムの最終アドレスに１を加えたもの、つまり、プログラムの初期DSまたはESの値の相対的な値(他のプログラムをロードできるオフセット)を設定し、そして割り込み27Hを実行します。これによりDOS/VはプログラムをDOSの拡張としてみなすので、そのプログラムは他のプログラムが実行される時に上書き(オーバーレイ)されません。この方法は、常駐のまま残さなければならない、ユーザーの書いた割り込みハンドラーのようなプログラムの場合のロードに有効です。

注:

1. この割り込みはメモリーの高アドレス端にロードされる、EXEプログラムで使用しないでください。
2. この割り込みは、割り込み20Hと同じような方法で割り込み22H, 23H, 24Hを復元します。そのため、それは永続的に常駐しているCtrl-Breakまたは重大エラーのハンドラー・ルーチンを導入するためには使用できません。
3. この方法によって常駐できるメモリーの最大サイズは64KBです。
4. 環境のコピーを含むブロックを終了前に割り振り解除すると、メモリーをもっと有効に使用することができます。これは、PSPの2CHに含まれるセグメントをESにロードし、ファンクション・コール49H(割り当てられたメモリーの解放)を発行することで行えます。

5. DOS/Vファンクション・コール4CHを使えば、終了プログラムが完了(またはエラー)コードをDOS/Vに渡すことが可能です。このコードはバッチ処理中で解釈することができます。

6. 常駐のまま終了のファンクション・コールは、自動的にファイルをクローズしません。

28H-2EH DOS/Vのために予約済み

これらの割り込みは、DOS/Vのために予約されています。

2FH多重割り込み

割り込み2FHは多重割り込みです。２つのプロセス間での一般的なインターフェースが定義されています。割り込み2FHを使用する特定のアプリケーション・プログラムは、特定の機能とパラメーターを定義します。

多重割り込みハンドラーごとに特定の多重番号が割り当てられます。多重番号は、AHレジスターで指定されます。ハンドラーが実行する特定の機能は、ALレジスターで指定されます。他のパラメーターは、必要に応じて他のレジスターに置かれます。ハンドラーは、割り込み2FHの割り込みベクトルにチェインされます。ハンドラーに多重番号を割り当てる方法は定義されていませんので、ひとつ選択しなければなりません。２つのアプリケーション・プログラムが同じ多重番号を選択した場合に衝突するのを避けるために、アプリケーション・プログラムに使用される多重番号はパッチできるようすべきです。

AH=00HからBFHまでの多重番号は、DOS/Vのために予約されています。アプリケーションは、C0HからFFHの多重番号を使用しなければなりません。

注: 割り込み2FHのチェインの時に、コードがDOS/Vをコールしたり、割り込み可能状態で実行する場合、そのコードは再入可能(reentrant)または再帰的(recursive)でなければなりません。

機能コード

AH=1

AH=１はPRINTの常駐部分です。

次の表はPRINTの常駐部分に対し、ALの中にコードを指定して、特定の機能を行うための機能コードを示しています。

機能コード	意味
0	導入状態の取得
1	ファイルの受渡し
2	ファイルの取り消し
3	すべてのファイルの取り消し
4	状況
5	状況の終了

PRINTエラー・コード

次の表は、プリントの常駐部分から返されるエラー・コードを示します。

エラー・コード	意味
1	無効な機能
2	ファイルが見つからない
3	パスが見つからない
4	オープンしたファイルが多過ぎる
5	アクセスが拒否された
6	待ち行列がいっぱい
9	使用中(Busy)
12	名前が長すぎる
15	無効なドライブ

AL=０導入された状態の取得

このコールは、すべての割り込み2FHハンドラーによって定義されなければなりません。それは、ハンドラーが存在するかどうか知るためにハンドラーのコールする側で使

用されます。エントリーはAL=０, AL=１です。リターンは、次に示すようにALレジスターが導入された状態を含みます。

AL=0 導入は許されているが導入されなかった
AL=1 導入は許されておらず、導入もされなかった
AL=FF 導入された

AL=1ファイルの受渡し

エントリーはAL=１, AH=１で、DS:DXが受渡しパケットを指します。受渡しパケットは、そのレベル(BYTE)とASCIIZ文字列へのポインター(オフセット・セグメント形式のDWORD)を含みます。DOS/Vに対するレベル値は０です。ASCIIZ文字列は、ドライブ、パス、印刷したいファイルのファイル名を含まなければなりません。ファイル名は、グローバル・ファイル名の文字列を含むことはできません。

AL=２ファイルの取り消し

エントリーは、AL=２, AH=１で、DS:DXは、取り消したいプリント・ファイルのASCIIZ文字列を指します。グローバル・ファイル名の文字列は、ファイル名の中で許されています。

AL=3すべてのファイルの取り消し

エントリーは、AL=３, AH=１です。

AL=4状況

このコールは、待ち行列を走査できるように印刷待ち行列のジョブを保留します。他のコードを発行するとジョブを解放します。エントリーは、AL=４, AH=１です。リターンは、DXレジスターがエラー数を含みます。エラー回数は、最後の文字を出力しようとする時にプリントで起こった連続した障害の数です。障害がなければ、数は０です。DS:DIは、印刷待ち行列を指します。印刷待ち行列は、連続するファイル名エントリーから構成されます。各エントリーは、64バイトの長さです。待ち行列の中の最初のエントリーは、現在印刷しているファイルです。待ち行列の最後は、最初の文字としてヌルを持つエントリーで示されます。

AL=5状況の終了

コールしてから待ち行列を解放するために、このコールを発行します。入力は、AL=５およびAH=１です。リターンでは、AXがエラーコードを含みます。返されるエラーコードの情報については、A-10ページの『PRINTエラー・コード』を参照してください。

AL=F8–FFはDOS/Vによって予約済み

AH=6

AH=６は、ASSIGNの常駐部分です。「導入状態の取得」機能(AL=０)がサポートされています。

AH=8H, 12H, 13HはDOS/Vによって予約済み

AH=10H

AH=10Hは、SHAREの常駐部分です。「導入状態の取得」機能(AL=０)がサポートされています。

AX = 1680H

おおくのアプリケーションはループ状態にあります。通常は、ユーザーからの何らかの入力や応答をアプリケーションが待っているときに、この状態になります。オペレーティング・システムにアプリケーションが待ち状態にあることを通知することは次のようなことに有用です。

- 電力制御ソフトウェアは、システム上のエネルギーをどのように保存するかについて判断を下すことができます。
- 多重タスク・ソフトウェアは、アイドルしているタスクにCPUサイクルを与えないようにして、時間を節約します。

DOS/Vアプリケーションがアイドルしていることを示すために、すべてのDOS/Vアプリケーションがこのapi（アプリケーション・プログラム・インターフェース）を使用するべきです。システム停止を防止するために、アプリケーションはInt 2fHベクターをチェックして、最初のアイドル呼出しが出される前にそれがゼロでないことを確認すべきです。APIがサポートされている環境では、Int 2fHはAL=0で戻ります。一方、APIがサポートされていない環境では、Int 2fHはAL=80H（無変更）で戻ります。通常はアプリケーションはこれらの戻り値に注意を払う必要はありません。

APIはブロッキングではありません。これは、APIを呼び出した後も、アプリケーションが続行されることを意味します。もしアプリケーションがアイドル状態のままであるなら、ユーザー・プログラムはこの割り込みを再度出すようにアイドル・ループを含んでいるべきです。

AH=B7H

AH=B7Hは、APPENDの常駐部分です。「導入状態の取得」機能(AL=０)がサポートされています。

AL=2 APPENDバージョンの取得。AX=FFFFHの場合、APPENDがロードされます。

AL=4 APPENDパス・ポインターの取得。リターンの、ES:DIが現在活動中のAPPENDパスを指します。

AL=6 APPEND機能状態の取得。

BXにAPPENDが現在使用可能であるか、そしてどんな機能が有効であるかを示すビット設定が返されます。

ビット	ビットがオンの時に有効な機能
0	APPENDが使用可能
13	/PATH
14	/E
15	/X

注: この機能の効力はAPPENDが使用不可であっても変わりません。

AL=7機能状態の設定

入力BXはすべての機能の新しい設定です。現在の機能状態を取得する推奨される手順は、希望するビットをオンまたはオフにし、そしてこのコールを機能状態の設定に使うことです。

AL=11H見つけた名前状態のリターン設定

AL=17のリクエストでは、処理システムの状態フラグが設定されます。もし、このフラグが設定されると、次のASCIIZがAPPENDを処理する割り込み21H内の3DH, 43Hまたは6CHファンクション・コールの場合、APPENDは、アプリケーション・プログラム・ファイル名バッファーから見つけた名前を返します。この名前は提供されたアプリケーション・プログラムとは異なるかもしれません。アプリケーション・プログラムは、見つけた名前のために十分な空間を用意しなければなりません。APPENDが割り込み21Hを処理した後、それは「見つけた名前のリターン」をリセットします。この処理の例を次に示します。

```
        MOV     AH,0B7H         ; APPEND指定
        MOV     AL,0            ; 導入状態の取得
        INT     2FH
        CMP     AL,0            ; APPENDは導入されたか?
        JE      NOT_INSTALLED

        MOV     AH,0B7H         ; APPEND指定
        MOV     AL,2            ; APPENDのバージョンの取得
        INT     2FH
        CMP     AX,-1           ; DOSのバージョン?
        JNE     APPEND

; 次の機能はDOS/VのAPPENDにおいてのみ有効

        MOV     AH,0B7H         ; APPENDを指定
        MOV     AL,4            ; APPENDパス・ポインターの取得
        INT     2FH

                                ; ES:DI = APPENDパスのアドレス
                                ; (バッファーは128の文字列長)

        MOV     AH,0B7H         ; APPENDを指定
        MOV     AL,6            ; 機能状態の取得
        INT     2FH
                                ; BX = 機能状態
                                ; 8000H = /Xがon
                                ; 4000H = /Eがon
                                ; 2000H = /PATHがon
                                ; 0001H = APPEND使用可能
                                ; offの場合APPENDパスのヌルと同様
                                ; どんなAPPENDの非状態発生でも設定

        MOV     AH,0B7H         ; APPENDを指定
        MOV     AL,7            ; 機能状態の設定
        MOV     BX,state        ; 新しい状態
        INT     2FH

        MOV     AH,0B7H         ; APPENDを指定
        MOV     AL,11H          ; 「見つけた名前状態リターン」の設定
        INT     2FH
```

2FHハンドラーの例

```
MYNUM           DB     X ; X = 特定の多重番号AH
INT_2F_NEXT     DD     ? ; チェイン位置
INT_2F:

ASSUME DS:NOTHING,ES:NOTHING,SS:NOTHING

  CMP     AH,MYNUM
  JE      MINE
  JMP     INT_2F_NEXT   ; 次の2FHハンドラーへのチェイン

MINE:

  CMP     AL,0F8H
  JB      DO_FUNC
  IRET                  ; 予約済み機能のIRET

DO_FUNC:

  OR      AL,AL
  JNE     NON_INSTALL   ; 導入取得をしない
                        ; 状態リクエスト
  MOV     AL,0FFH       ; 「私はここ」と言う
  IRET                  ; すべて終了

NON_INSTALL:
⋮
```

ハンドラーの導入

次の例は、ハンドラーを導入するために必要な機能を含みます。

```
   MOV    AH,MYNUM
   XOR    AL,AL
   INT    2FH                       ; すでに導入済みかを問い合わせる
   OR     AL,AL
   JZ     OK_INSTALL

BAD_INSTALL:                        ; ハンドラーは導入済み



OK_INSTALL:                         ; 自分のハンドラーを導入

   MOV    AL,2FH
   MOV    AH,GET_INTERRUPT_VECTOR
   INT    21H                        ; 多重ベクトルの取得
   MOV    WORD PTR INT_2F_NEXT+2,ES
   MOV    WORD PTR INT_2F_NEXT,BX
   MOV    DX,OFFSET INT_2F
   MOV    AL,2FH
   MOV    AH,SET_INTERRUPT_VECTOR
   INT    21H                        ; 多重ベクトルの設定
⋮
```

30H-3FH DOS/Vのために予約済み

これらの割り込みはDOS/Vのために予約されています。

付録B. DOS/Vファンクション・コール

番号	機能名
00H	プログラムの終了
01H	コンソール入力(エコーあり)
02H	ディスプレイ出力
03H	補助入力
04H	補助出力
05H	プリンター出力
06H	直接コンソール入出力
07H	直接コンソール入力（エコーなし）
08H	コンソール入力（エコーなし）
09H	文字列表示
0AH	バッファー付きキーボード入力
0BH	標準入力状況のチェック
0CH	キーボード・バッファーの消去およびキーボード機能の呼び出し
0DH	ディスク・リセット
0EH	ディスクの選択
0FH	ファイルのオープン
10H	ファイルのクローズ
11H	最初のエントリーの探索
12H	次のエントリーの探索
13H	ファイルを削除
14H	順次読み取り
15H	順次書き込み
16H	ファイルの作成
17H	ファイル名の変更
18H	DOS/Vにより予約済み
19H	現行ディスク
1AH	ディスク転送アドレス(DTA)の設定
1BH	アロケーション(割り当て)テーブル情報
1CH	指定装置の割り振り
1DH	DOS/Vにより予約済み
1EH	DOS/Vにより予約済み
1FH	DOS/Vにより予約済み
20H	DOS/Vにより予約済み
21H	ランダム読み取り
22H	ランダム書き込み
23H	ファイル・サイズ
24H	相対レコード・フィールドの設定
25H	割り込みベクトルの設定
26H	新しいプログラム・セグメントの作成

27H ランダム・ブロック読み取り
28H ランダム・ブロック書き込み
29H ファイル名解析
2AH 日付の取得
2BH 日付の設定
2CH 時刻の取得
2DH 時刻の設定
2EH ベリファイ・スイッチの設定と解除
2FH ディスク転送アドレス(DTA)の取得
30H DOS/Vバージョン番号の取得
31H 常駐のままプロセス終了
32H DOS/Vにより予約済み
33H システム値の取得および設定
34H DOS/Vにより予約済み
35H 割り込みベクトルの取得
36H ディスク空き領域の取得
37H DOS/Vにより予約済み
38H 国別情報の取得または設定
39H サブディレクトリーの作成(MKDIR)
3AH サブディレクトリーの削除(RMDIR)
3BH 現行ディレクトリーの変更(CHDIR)
3CH ファイルの作成(CREAT)
3DH ファイルのオープン
3EH ファイル・ハンドルのクローズ
3FH ファイルまたは装置から読み取り
40H ファイルまたは装置への書き込み
41H 指定されたディレクトリーからのファイルの削除(UNLINK)
42H ファイル読み書きポインターの移動(LSEEK)
43H ファイル・モードの変更(CHMOD)
44H 装置のための入出力制御(IOCtl)
45H ファイル・ハンドルの複製(DUP)
46H ファイル・ハンドルの強制複製(FORCDUP)
47H 現行ディレクトリーの取得
48H メモリーの割り当て
49H 割り当てられたメモリーの解放
4AH 割り当てられたメモリー・ブロックの変更(SETBLOCK)
4BH プログラムのロードまたは実行(EXEC)
4CH プロセスの終了(EXIT)
4DH サブプロセスからのリターン・コードの取得(WAIT)
4EH 最初に一致するファイルを見つける(FIND FIRST)
4FH 次に一致するファイルを見つける(FIND NEXT)
50H DOS/Vにより予約済み
51H DOS/Vにより予約済み
52H DOS/Vにより予約済み

53H	DOS/Vにより予約済み
54H	ベリファイの設定状況の取得
55H	DOS/Vにより予約済み
56H	ファイル名の変更
57H	ファイルの日付および時刻の取得と設定
58H	DOS/Vにより予約済み
59H	拡張エラーの取得
5AH	ユニーク・ファイルの作成
5BH	新しいファイルの作成
5CH	ファイル・アクセスのロックまたはロックの解除
5DH	DOS/Vにより予約済み
5E00H	機械名の取得
5E02H	プリインター・セットアップの設定
5E03H	プリンター・セットアップの取得
5F02H	リダイレクション・リスト・エントリーの取得
5F03H	装置のリダイレクト
5F04H	リダイレクションの取り消し
60H	DOS/Vにより予約済み
61H	DOS/Vにより予約済み
62H	PSPアドレスの取得
6300H	DBCSベクター情報の取得
64H	DOS/Vにより予約済み
65H	拡張国別情報の取得
66H	グローバル・コード・ページの取得と設定
67H	ハンドル数の設定
68H	ファイルのコミット
69H	DOS/Vにより予約済み
6AH	予約済み
6BH	DOS/Vにより予約済み
6CH	拡張オープンおよび作成

DOS/Vファンクション・コールの使用

多くのファンクション・コールは、それらにレジスターで渡される入力が必要です。適当なレジスターの値を設定した後、次のいずれかの方法によってファンクション・コールを発行します。

- 推奨される方法は、AHにファンクション番号を入れて、割り込み21Hを発行する方法です。
- AHに機能番号を入れ、プログラム・セグメントプリフィクス内のオフセットをコールして50H実行する方法です。

プログラム・コード・フラグメント

この章で述べられるファンクション・コールの各々について、入力、出力、使用方法は小さなプログラム・コード・フラグメントを使って説明します。これらのフラグメントは、アセンブラー言語で書かれています。

.COMプログラム

プログラムは.COMプログラムでなく、.EXEプログラムであると仮定して説明します。.COMプログラムが必要ならば、次の命令のいずれかを除いてください。

```
  MOV ES,SEG –
or
  MOV DS,SEG –
```

注:

1. 幾つかのFCBファンクション・コールは、無効な文字列(0DH-29H)を許可しません。
2. コロンで終わる装置名は使用できません。
3. AXレジスターの内容は、ファンクション・コールによって変更されます。エラー・コードがAXに返されなくとも、AXは変更される可能性があります。

DOS/Vレジスター

DOS/Vは、割り込みやファンクション・コールを実行する時に次のレジスター、ポインター、フラグを使用します。

レジスター定義	汎用レジスター
AX AH AL	アキュムレーター (16ビット) アキュムレーター上位バイト(８ビット) アキュムレーター下位バイト(８ビット)
BX BH BL	ベース(１６ビット) ベース上位バイト(８ビット) ベース下位バイト(８ビット)
CX CH CL	カウント(16ビット) カウント上位バイト(８ビット) カウント下位バイト(８ビット)
DX DH DL	データ(16ビット) データ上位バイト(８ビット) データ下位バイト(８ビット)
フラグ	OF,DF,IF,TF,SF,ZF,AF,PF,CF

レジスター定義	ポインター
SP	スタック・ポインター(16ビット)
BP	ベース・ポインター(16ビット)
IP	インストラクション(命令)ポインター(16ビット)

レジスター定義	セグメント・レジスター
CS	コード・セグメント(16ビット)
DS	データ・セグメント(16ビット)
SS	スタック・セグメント(16ビット)
ES	エクストラ・セグメント(16ビット)

レジスター定義	インデックス(索引)レジスター
DI	デスティネーション(宛先)インデックス(16ビット)
SI	ソース・インデックス(16ビット)

レジスター番号規定

各レジスターは、16ビットの長さで上位および下位バイトに分けられます。各バイトは８ビットの長さです。ビットには、右から左に番号がつけられています。下位バイトは０から７で、上位バイトが８から15を含みます。下の表は、各ビットに割り当てられた16進値を示します。

	上位バイト	下位バイト
ビット	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
16進値	8 4 2 1 8 4 2 1	8 4 2 1 8 4 2 1

DOS/V内部スタック

DOS/Vが制御を得ると、それは内部スタックに切り替わります。ユーザー・レジスターは、特別のリクエストで示されるように、情報がリクエスト機能に戻されるまで保存されます。ユーザー・スタックは、割り込みシステムを受け入れるのに十分である必要があります。ユーザー・スタックは、ユーザーの必要とするものにさらに200Hを加えることが推奨されます。

エラーに対する応答

ハンドルファンクション・コールは、キャリー・フラグの設定によってエラーを報告し、AXでエラー・コードを返します。FCBファンクション・コールは、ALでFFHを返すことによってエラーを報告します。

拡張エラー・サポート(59H)は、共通のエラー・コード・セットと特定のエラー情報、たとえば、エラーの分類、位置、推奨される対処などを提供します。多くの重大な状態では、アプリケーション・プログラムはエラー・コードを解析し、特定の動作がとれます。推奨される動作は、エラー・コードを理解しないプログラム用です。プログラムは、割り込み24H重大なエラー・ハンドラーからと割り込み21Hファンクション・コールを発行した後の両方で、拡張エラー・サポートを利用できます。特定のエラー・コードに対しコード化してはいけません。

拡張エラー・コード

16進コード	**10進コード**	**意味**
01H	1	無効な機能番号
02H	2	ファイルが見つからない
03H	3	パスが見つからない
04H	4	オープンしたファイルが多すぎる(ハンドルが残っていない)
05H	5	アクセスが拒否された
06H	6	無効なハンドル
07H	7	メモリー制御ブロックが破壊された
08H	8	メモリー不足
09H	9	無効なメモリー・ブロック・アドレス
0AH	10	無効な環境
0BH	11	無効な形式
0CH	12	無効なアクセス・コード
0DH	13	無効なデータ
0EH	14	予約済み
0FH	15	無効なドライブが指定された
10H	16	現行ディレクトリーを削除しようとした

16進コード	10進コード	意味
11H	17	同じデバイスではない
12H	18	これ以上ファイルなし
13H	19	書き込み禁止ディスケットに書き込もうとした
14H	20	未定義ユニット
15H	21	ドライブの準備ができていない
16H	22	未定義コマンド
17H	23	CRC (Cyclic redundancy check)—ディスケットの一部が不良
18H	24	不良リクエスト構造長
19H	25	シーク・エラー
1AH	26	未定義メディア・タイプ
1BH	27	セクターが見つからない
1CH	28	プリンター用紙切れ
1DH	29	書き込み失敗
1EH	30	読み取り失敗
1FH	31	一般的な失敗
20H	32	共用違反
21H	33	ロック違反
22H	34	無効なディスク交換
23H	35	FCBが使用不能
24H	36	共用バッファーあふれ
25H	37	DOS/Vにより予約済み
26H	38	ファイル操作を完了できない
27H–31H	39–49	DOS/Vのために予約済み
50H	80	ファイルが存在する
51H	81	予約済み

16進コード	10進コード	意味
52H	82	ディレクトリー・エントリーができない
53H	83	INT24Hで障害
54H	84	リダイレクションが多すぎる
55H	85	リダイレクションが重複している
56H	86	パスワードが無効
57H	87	パラメーターが無効
58H	88	ネットワーク・データ障害
59H	89	ネットワークによってサポートされない機能
5AH	90	リクエストされたシステム・コンポーネントが導入されていない

拡張エラー・コード

16進コード	10進コード	意味
32H	50	ネットワーク・リクエストがサポートされていない
33H	51	リモート・コンピューターが聴取(Listening)していない
34H	52	ネットワーク名重複
35H	53	ネットワーク・パスが見つからない
36H	54	ネットワーク使用中(Busy)
37H	55	ネットワーク装置がすでに存在しない
38H	56	NETBIOSコマンドの限界を超えている
39H	57	システム・エラー、NETBIOSエラー
3AH	58	ネットワークからの間違った応答
3BH	59	予期しないネットワーク・エラー
3CH	60	互換性のないリモート・アダプター
3DH	61	印刷(プリント)待ち行列がいっぱい

16進コード	10進コード	意味
3EH	62	印刷ファイルのための空間が不足
3FH	63	印刷ファイルが取り消された
40H	64	ネットワーク名が削除された
41H	65	アクセスが拒否された
42H	66	ネットワーク装置タイプが間違っている
43H	67	ネットワーク名が見つからない
44H	68	ネットワーク名の限界を超えている
45H	69	NETBIOSセッション限界を超えている
46H	70	共用を一時休止
47H	71	ネットワーク・リクエストが受け付けられない
48H	72	印刷またはディスク・リダイレクション休止
49H–4FH	73–79	予約済み

エラー分類

16進値	10進値	意味
01H	1	**リソース(資源)が尽きた:** たとえば空間またはチャンネルなどの不足
02H	2	**一時的状態:** 時間とともに解決すると思われる状況。これは、エラー状態ではありません。たとえばロックされたファイルのような一時的状態を意味します。
03H	3	**許可:** 許可の問題。
04H	4	**内部:** システム・ソフトウェアの内部エラー。ユーザーやシステムの障害よりもむしろシステム・ソフトウェアに伴う問題。
05H	5	**ハードウェアの障害:** ユーザー・プログラムの失敗ではない重大な問題。
06H	6	**システムの障害:** システム・ソフトウェアの重大な障害。たとえばコンフィギュレーション・ファイルが見つからないとか間違っているといったような、ユーザーの失敗ではない問題。

16進値	10進値	意味
07H	7	**アプリケーション・プログラム・エラー:** 矛盾するリクエスト。
08H	8	**見つからない:** ファイルまたは項目が見つからない。矛盾するリクエスト。
09H	9	**間違ったフォーマット:** 無効なフォーマットまたはタイプのファイルまたは値。不適当。
0AH	10	**ロックされている:** ロックされたファイルまたは項目。
0BH	11	**メディア:** メディア障害。たとえば間違ったディスク、CRCエラー、または欠陥メディアなど。
0CH	12	**すでに存在している:** 重複エラー。たとえばすでに存在する機器名の宣言など。
0DH	13	**未定義:** 分類が存在しないかまたは不適当。

動作

16進コード	10進コード	意味
01H	1	**再試行:** 2、3回再試行し、それからプログラムを続行するか、または終了するかを決定するようにユーザーにプロンプトを出す。
02H	2	**遅延(delay)再試行:** ポーズ後、幾度か再試行し、それからプログラムを続行するか、または終了するかを決定するようにユーザーにプロンプトを出す。
03H	3	**ユーザー:** もし、入力がユーザーによるものであれば、再入力を促す。しかし、たとえば間違ったドライブ文字または間違ったファイル名指定のようなエラーは、プログラム自身で起こる可能性がある。
04H	4	**中止:** アプリケーションの中止と消去。アプリケーションは続行できないが、システムは通常状態でアプリケーションを中止するのに安全。
05H	5	**即時終了:** レジスターを消去せずに、すぐにアプリケーションを中止する。ファイルのクローズやインデックスの更新には使えない。ただしできるだけ早く終了する。

16進コード	10進コード	意味
06H	6	**強行:** 無視
07H	7	**ユーザー介入後に再試行:** ユーザーはディスケットを取り出し、別のディスケットを入れるといった動作をする必要がある。それから、操作を再試行する。

位置

16進値	10進値	意味
01H	1	**未定義:** 特定化されていない。不適当。
02H	2	**ブロック装置:** ランダム・アクセスの大容量ディスクに関連。
03H	3	**ネット:** ネットワークに関連。
04H	4	**シリアル装置:** シリアル装置に関連。
05H	5	**メモリー:** ランダム・アクセス・メモリーに関連。

00H —
プログラムの終了

目的

プログラムの実行を停止します。

例

```
MOV     AH,00H   ; ファンクション・コール–プログラムの終了
INT     21H      ; DOSにリクエストを発行
                 ; リターンなし
```

解説

終了、Ctrl-Break、重大エラー出口アドレスは、プログラム・セグメント・プリフィックスに保存された値から、終了プログラムのエントリーの値に復元されます。すべてのファイル・バッファーは、フラッシュ(空に)され、プロセスによってオープンされたハンドルは、クローズされます。長さの変更されたファイルおよびクローズされなかったファイルは、どれもディレクトリーにきちんと記録されません。制御は終了アドレスに移ります。このコールは、割り込み20Hと全く同じ機能を実行します。CSレジスターがこの機能をコールする前に、そのプログラム・セグメントプリフィックス制御ブロックのセグメント・アドレスを含むようにすることは、プログラム側の責任です。

機能4CH—プロセスの終了が、プログラムを終了するために推奨される方法です。

01H —
コンソール入力(エコーあり)

目的

標準入力装置で読み込まれる１文字を(１文字が入力されるまで)待ちます。それから、標準出力装置へ文字をエコーし、ALでその文字を返します。

例

```
       MOV     AH,01H          ; ファンクション・コール–キーボード入力
       INT     21H             ; DOSにリクエストを発行
       MOV     Char,AL         ; 文字を保存
       CMP     AL,0            ; 拡張文字か?
       JNE     Normal_Char     ; いいえ
       MOV     AH,01H          ; キーボード入力INT
       INT     21H             ; DOSにリクエストを発行
       MOV     ExtChar,AL      ; 拡張文字の保存
Normal_Char:

 -----

 Character     LABEL   WORD    ; 完全な文字
 Char          DB      ?       ; 文字バッファー
 ExtChar       DB      ?       ; 文字バッファー
```

解説

文字は、Ctrl-Breakかどうかチェックされます。Ctrl-Breakが検知されると割り込み23Hが実行されます。

ファンクション・コール01Hで、拡張アスキー・コードは２つのファンクション・コールを必要とします。最初のコールは、00Hを返します。これは次のコールが拡張コードを返すことを示すものです。

02H —
ディスプレイ出力

目的

標準出力装置にDLレジスター内の文字を出力します。

例

```
    MOV     AH,02H          ; ファンクション・コール–ディスプレイ出力
    MOV     DL,Char         ; 表示する文字の取得
    INT     21H             ; DOSにリクエストを発行

-----

CHAR        DB         ?    ; 文字バッファー
```

解説

DLレジスター内の文字がバックスペース(08)の場合、カーソルが(文字を消すことなく)左に１つ移動します。Ctrl-Breakが出力の後に検出されると割り込み23Hが実行されます。

03H —
補助入力

目的

標準補助入力装置からの１文字を待ちます。それから、ALで文字を返します。

例

```
        MOV     AH,03H          ; ファンクション・コール–補助入力
        INT     21H             ; DOSにリクエストを発行
        MOV     Char,AL         ; 文字の保存

-------

CHAR    DB      ?               ; 文字バッファー
```

解説

補助(AUX)サポートはバッファーなしで、非割り込み駆動型です。

開始の時にDOS/Vは、第１補助ポートをボーレート2400、パリティー無し、ストップビット１、データ８ビットに初期設定します。

補助ファンクション・コール(03Hと04H)は、状況またはエラー・コードを返しません。より大きな制御のためには、ROM BIOSルーチン(割り込み14H)を使用するか、AUXデバイス・ドライバーを書き込んでIOCtlを使用するようにしてください。

04H —
補助出力

目的

標準補助出力装置にDL内の文字を出力します。

例

```
        MOV     AH,04H          ; ファンクション・コール–補助出力
        MOV     DL,Char         ; 出力文字を取得
        INT     21H             ; DOSにリクエストを発行
                                ; 何も返らない

-------

CHAR    DB      ?               ; 文字バッファー
```

解説

DL内の文字がバックスペース(08)の場合、カーソルが(文字を消すことなく)左に１つ移動します。Ctrl-Breakが出力の後に検知された場合、割り込み23Hが実行されます。

05H —
プリンター出力

目的

標準プリンター装置にDL内の文字を出力します。

例

```
        MOV     AH,05H          ; ファンクション・コール–プリンター出力
        MOV     DL,Char         ; 出力文字の取得
        INT     21H             ; DOSにリクエストを発行
                                ; 何も返らない

--------

CHAR    DB        ?             ; 文字バッファー
```

06H —
直接コンソール入出力

目的

１文字の準備ができていれば、標準入力装置から１文字を読み取る、または標準出力装置に１文字を出力します。

例

```
 In_loop:
   MOV    AH,06H         ; ファンクション・コール–直接コンソール入出力
   MOV    DL,-1          ; 0FFH入力のため
   INT    21H            ; DOSにリクエストを発行
   JZ     In_loop        ; まだ文字の入力がない
   MOV    Char,AL        ; 文字の保存
   CMP    AL,0           ; 拡張文字か?
   JNE    Normal_Char    ; いいえ
   MOV    AH,07H         ; ファンクション・コール–キーボード入力
   INT    21H            ; DOSにリクエストを発行
   MOV    ExtChar,AL     ; 拡張文字の保存
Normal_Char:

 or

   MOV    AH,06H         ; ファンクション・コール–直接コンソール入出力
   MOV    DL,Char        ; 表示文字の取得(0FFHではない)
   INT    21H            ; DOSにリクエストを発行

 -----

 Character    LABEL  WORD ; 完全な文字
 Char          DB       ? ; 文字バッファー
 ExtChar       DB       ? ; 文字バッファー
```

解説

DLがFFHの場合、ALは０フラグをクリアし、標準入力装置からの入力文字の準備ができていれば、１入力文字をもって返ります。１文字が準備されていない場合は、０フラグが設定されます。

DLがFFHでない場合、DLは標準出力装置へ出力する有効な文字を持っているとみなされます。この機能は、Ctrl-BreakまたはCtrl-ページ印刷に対してチェックはされません。

ファンクション・コール06Hでは、拡張アスキー・コードは２つのファンクション・コールを必要とします。最初のコールは、00Hを返します。これは次のコールが拡張コードを返すことを示すものです。

07H —
直接コンソール入力(エコーなし)

目的

標準入力装置で読み込まれる１文字を(１文字が入力されるまで)待ちます。それから、ALでその文字を返します。

例

```
  MOV   AH,07H       ; ファンクション・コール–直接コンソール出力(エコーなし)
  INT   21H          ; DOSにリクエストを発行
  MOV   Char,AL      ; 文字の保存
  CMP   AL,0         ; 拡張文字 ?
  JNE   Normal_Char  ; いいえ
  MOV   AH,07H       ; ファンクション・コール–直接コンソール出力(エコーなし)
  INT   21H          ; DOSにリクエストを発行
  MOV   ExtChar,AL   ; 拡張文字の保存
Normal_Char:

  -----

  Character     LABEL   WORD    ; 完全な文字
  Char          DB      ?       ; 文字バッファー
  ExtChar       DB      ?       ; 文字バッファー
```

解説

ファンクション・コール06Hのように文字のチェックはしません。

ファンクション・コール07Hでは、拡張アスキー・コードは２つのファンクション・コールを必要とします。最初のコールは、00Hです。これは、次のコールが拡張コードを返すということを示すものです。

08H —
コンソール入力(エコーなし)

目的

標準入力装置で読み込まれる１文字を(１文字が入力されるまで)待ちます。それから ALでその文字を返します。

例

```
 MOV  AH,08H     ; ファンクション・コール–コンソール入力(エコーなし)
 INT  21H        ; DOSにリクエストを発行
 MOV  Char,AL    ; 文字の保存
 CMP  AL,0       ; 拡張文字か？
 JNE  Normal_Char; いいえ
 MOV  AH,08H     ; ファンクション・コール-コンソール入力(エコーなし)
 INT  21H        ; DOSにリクエストを発行
 MOV  ExtChar,AL ; 拡張文字を保存
Normal_Char:

  -----

 Character     LABEL   WORD    ; 完全な文字
 Char          DB      ?       ; 文字バッファー
 ExtChar       DB      ?       ; 文字バッファー
```

解説

文字はCtrl-Breakのチェックをされます。Ctrl-Breakが検知されると割り込み23Hが実行されます。

ファンクション・コール08Hでは、拡張アスキー・コードは２つのファンクション・コールを必要とします。最初のコールは、00Hです。これは次のコールが拡張コードを返すということを示すインジケーターです。

09H —
文字列表示

目的

標準出力装置に文字列中の文字を送ります。

例

```
MOV  AX,SEG String
MOV  DS,AX            ; 文字列を設定
MOV  DX,OFFSET String
MOV  AH,09H           ; ファンクション・コール-文字列出力
INT  21H              ; DOSにリクエストを発行

 -----

 String       DB      "この文字列は最初の＄で終わります"
              DB      0DH,0AH
              DB      "$"
```

解説

メモリーの文字列は$(24H)で終了していなければなりません。文字列の各文字は、ファンクション・コール02Hと同じ形式で標準出力装置に出力されます。

アスキー・コード0DHと0AHは、それぞれ復帰と改行を表しています。

0AH —
バッファ付きキーボード入力

目的

標準入力装置から文字を読み取り、それらを第３バイトから始まるバッファーに置きます。

例

```
MOV  AX,SEG Buffer
MOV  DS,AX              ; バッファーを返すDS:DXを設定
MOV  DX,OFFSET Buffer
MOV  AH,0AH             ; ファンクション・コール-バッファー付き
                        ; キーボード入力
INT 21H                 ; DOSにリクエストを発行

------

Buffer   DB   128       ; 入力の最大長
CurLen   DB   ?         ; 文字入力の数
                        ; (リターン(0DH)を除く)
CurText  DB   128 DUP(?) ; 128文字まで許す
```

解説

入力バッファーの第１バイトは、バッファーが保持できる文字数を指定します。この値は０にできません。Enterが読み取られるまで、標準入力装置からの読み取りと、バッファーへの格納を続けます。バッファーが保持できる文字の最大数よりひとつ少ない文字まで埋まると、その後タイプされた文字はすべて無視され、Enterが読み込まれるまでベルを鳴らします。バッファーの第２バイトは、受け取られる文字数を設定します。ただし常に最後の文字である復帰(0DH)は除きます。

0BH —
標準入力の状況チェック

目的

標準入力装置から文字が入力可能かどうかをチェックします。

例

```
In_LOOP:
  MOV     AH,0BH    ; ファンクション・コール—入力チェック
  INT     21H       ; DOSにリクエストを発行
  CMP     AL,-1     ; 0FFHは文字が入力可能を示す
  JNE     In_LOOP
```

解説

文字が標準入力装置から入力されていればALはFFHです。そうでなければALは未定義です。Ctrl-Breakが検出されると割り込み23Hが実行されます。

0CH —
キーボード・バッファーの消去およびキーボード機能の呼び出し

目的

標準入力装置から以前にタイプされた文字を消去し、それからAL内のファンクション・コール番号を実行します。

例

```
MOV    AH,0CH         ; ファンクション・コール—キーボード・バッファー
                      ; の消去およびファンクション・コール
MOV    AL,Function    ; 実行するファンクション・コール
                      ; (01H, 06H, 07H, 08H, と0AHだけ許されています。)
INT    21H            ; DOSにリクエストを発行
                      ; 出力は選択されたファンクション・コールによります
```

0DH —
ディスク・リセット

目的

修正されたファイル・バッファーをディスクに書き込みます。その後はすべてのバッファーが再利用できます。

例

```
MOV     AH,0DH          ; ファンクション・コール—ディスク・リセット
INT     21H             ; DOSにリクエストを発行
                        ; リターンなし
```

解説

ディスク・ディレクトリーを正しく更新するために、すべてのオープンしたファイルをクローズするまたはコミットする必要があります。

0EH —
ディスクの選択

目的

DLで指定されたドライブ(０=A,１=Bなど)を省略時ドライブとして(有効ならば)選択します。

例

```
MOV     AH,0EH          ; ファンクション・コール–ディスクの選択
MOV     DL,Drive        ; 選択するドライブ      (0=A:, 1=B:, ...)
INT     21H             ; DOSにリクエストを発行
MOV     LastDrive,AL    : 最大ドライブ数の保存 (1=A:, 2=B:, ...)
MOV     AH,19H          ; ファンクション・コール–現行ディスクの取得
INT     21H             ; DOSにリクエストを発行
CMP     AL,DL           ; リクエストされたドライブ
JNE     Error           ; エラーなし！

-----

Drive          DB       ; 選択する新しいドライブ
LastDrive      DB       ; 有効ドライブの最大値
```

解説

ディスケットと固定ディスク・ドライブを含む固有のドライブ文字の合計は、AL内で返される値で参照できます。ALの値は、CONFIG.SYS内のLASTDRIVEの値、または導入された装置の合計のどちらか大きい方と等しい値です。DOS/Vでは、５がALで返される最小値です。もし、システムが１つのディスケット・ドライブしか持たない場合でも、論理ドライブAとBを持つというシステムの考え方にもとづいて、２として数えられます。

0FH —
ファイルのオープン

目的

名前の付けられたファイルを現行ディレクトリーで探し、それが見つからないとALはFFHを返します。ファイルが見つかればALは00Hを返し、FCBは下に示すように埋められます。

例

```
MOV  AX,SEG FCB     ; FCBパラメータ・ブロックへアドレス
MOV  DS,AX
MOV  DX,OFFSET FCB
MOV  AH,0FH         ; ファンクション・コール-FCBオープン
INT  21H            ; DOSにリクエストを発行

 -----

 FCB          LABEL   BYTE
 Drive        DB      0               ; ドライブ(0=現行,1=A, 2=B, ...)
 FName        DB      "FILENAME"      ; ファイル名(ブランクで埋まる)
 Ext          DB      "EXT"           ; ファイル・エクステンション
                                      ; (ブランクで埋まる)
              DB      25 DUP(0)       ;  DOS/Vにより埋められる
```

解説

ファイルがオープンしている場合、ALは00Hです。

ファイルがオープンしていない場合、ALはFFHです。

実際のエラー状態を決定するにはファンクション・コール59H (拡張エラーの取得)を使用してください。

ドライブ・コードが０(省略時ドライブ)の場合、使用されている実際のドライブ(１=A, ２=Bなど)に変更されます。これにより、このファイル上の引き続く操作に干渉せずに省略時ドライブを変更できます。現行のブロック・フィールド(FCBバイトC-D)は０に設定されます。操作されるレコード・サイズ(FCBバイトE-F)は、システム・省略値の80Hに設定されます。ファイルのサイズと日付は、ディレクトリーから得られた情報からFCBに設定されます。レコード・サイズ(FCBバイトE-F)の省略値を変更したり、ランダム・レコード・サイズ、現行レコード・フィールドの設定ができます。ファイルをオープン後、ディスク操作の前に、これらの動作を行ってください。

ファイルは互換モードでオープンされます。互換性モードの情報については、ファンクション・コール3DHを参照してください。

10H —
ファイルのクローズ

目的

ファイルをクローズします。

例

```
MOV  AX,SEG FCB         ; FCBパラメータのブロックへアドレス
MOV  DS,AX
MOV  DX,OFFSET FCB
MOV  AH,10H             ; ファンクション・コール-FCBクローズ
INT  21H                ; DOSにリクエストを発行
CMP  AL,0               ; ファイルはクローズか?
JNE  Error              ; エラーなし!

-----

FCB           LABEL   BYTE
;前の操作で設定された項目
```

解説

ファイルがクローズしている場合、ALは00Hです。

ファイルがクローズしていない場合、ALはFFHです。

実際のエラー・コードを決定するためにファンクション・コール59H (拡張エラーの取得)を使用してください。

このファンクション・コールは、ファイルの書き込み後に、オープンしているファイルに対し実行されなければなりません。すべてのファイルにこれを使用することを強くお勧めします。現行ディレクトリーの正しい位置にファイルが見つからない場合、ディスクが交換されたとみなして、ALがFFHを返します。そうでなければディレクトリーは、FCBに状況を反映するために更新され、そのファイルのバッファーはフラッシュされ、ALは00Hを返します。

11H —
最初のエントリーの探索

目的

最初に一致するファイル名を現行ディレクトリーで探索します。

例

```
MOV  AX,SEG DTA    ; 見つかったファイル情報のバッファーへアドレス
MOV  DS,AX
MOV  DX,OFFSET DTA
MOV  AH,1AH        ; ファンクション・コール-DTAアドレスの設定
INT  21H           ; DOSにリクエストを発行
MOV  AX,SEG FCB    ; FCBパラメーター・ブロックへアドレス
MOV  DS,AX
MOV  DX,OFFSET FCB
MOV  AH,11H        ; ファンクション・コール-FCB最初のエントリーの探索
INT  21H           ; DOSにリクエストを発行
CMP  AL,0          ; ファイルが見つかったか?
JNE  Error         ; エラーなし!

-----

FCB     LABEL   BYTE
Fdrive  DB      0                ; ドライブ(0=現行, 1=A, 2=B, ...)
Fname   DB      "FILENAME"       ; ファイル名(ブランクで埋まる、?も使用可)
Fext    DB      "EXT"            ; ファイル・エクステンション(ブランク
                                 ; で埋まる、?も使用可)
        DB      25 DUP(0)        ; DOSにより埋められる

DTA     LABEL   BYTE
Ddrive  DB      ?                ; ドライブ
Dname   DB      "????????"       ; ファイル名(ブランクで埋まる)
Dext    DB      "???"            ; ファイル・エクステンション(ブランク
                                 ; で埋まる)
        DB      25 DUP(0)        ; DOS/Vにより埋められる
```

解説

ファイルが見つかった場合、ALは00Hです。

ファイルが見つからなかった場合、ALはFFHです。

実際のエラー状態を決定するためにはファンクション・コール59H(拡張エラーの取得)を使用してください。

現行ディスク・ディレクトリーでは、最初に一致するファイル名が探索されます。何も見つからない場合は、ALはFFHを返します。*や?のグローバル・ファイル名文字は、ファイル名とエクステンションに含めることができます。一致するファイル名が見つかるとALは00Hを返し、ディスク転送アドレスの位置が次のように設定されます。

- 探索のために提供されるFCBが拡張FCBだった場合、ディスク転送アドレスの最初のバイトはFFHに設定され、その後に５バイトの０が続き、探索FCBからの属性バイト、使用されているドライブ番号(１=A, ２=Bなど)、それからディレクトリー・エントリーの32バイトが続きます。このように、ディスク転送アドレスは、探索FCBと同じ探索属性を持つ有効な非オープンの拡張FCBを含みます。

- 探索のために提供されるFCBが標準FCBだった場合、最初のバイトは使用されているドライブ番号(１=A, ２=B)に設定され、次の32バイトは、一致するディレクトリー・エントリーを含みます。このように、ディスク転送アドレスは、有効な非オープンの通常FCBを含みます。

注:

拡張FCBが使用される場合、次の探索パターンが使用されます。

1. 属性が０の場合、通常ファイル・エントリーだけが見つけられます。ボリューム・ラベル、サブディレクトリー、隠しファイルおよびシステム・ファイルに対するエントリーは返されません。

2. 属性フィールドが、隠しファイル、システム・ファイル、またはディレクトリー・エントリーに対して設定される場合、すべてを含んだ探索となります。すべての通常のファイル・エントリーに加えて、指定された属性に一致するすべてのエントリーが返されます。ボリューム・ラベルを除くすべてのディレクトリー・エントリーを見るには、属性バイトを隠しファイル+システム+ディレクトリー(３ビットすべてオン)に設定するとよいでしょう。

3. 属性フィールドがボリューム・ラベルに設定されている場合、排他的探索と考えられます。そしてボリューム・ラベル・エントリーのみが返されます。

12H —
次のエントリーの探索

目的

次に一致するファイル名を現行ディレクトリーで探索します。

例

```
MOV       AX,SEG DTA    ; 見つかったファイル情報のバッファーにアドレス
MOV       DS,AX
MOV       DX,OFFSET DTA
MOV       AH,1AH        ; ファンクション・コール-DTAアドレスの設定
INT       21H           ; DOSにリクエストを発行
MOV       AX,SEG FCB    ; FCBパラメーター・ブロックにアドレス
MOV       DS,AX
MOV       DX,OFFSET FCB
MOV       AH,12H        ; ファンクション・コール-次のFCBの探索
INT       21H           ; DOSにリクエストを発行
CMP       AL,0          ; ファイルが見つかった?
JNE     Error           ; エラーなし!

-----

FCB            LABEL   BYTE
; 最初のFCBの探索と同様に設定

DTA            LABEL   BYTE
Drive          DB      ?              ; ドライブ
Fname          DB      "????????"     ; ファイル名(ブランクで埋まる)
Ext            DB      "???"          ; ファイル・エクステンション(ブランク
                                      ; で埋まる
               DB      25 DUP(0)      ; DOS/Vにより埋められる
```

解説

ファイルが見つかった場合、ALは00Hです。

ファイルが見つからなかった場合、ALはFFHです。

実際のエラー状態を決定するためにはファンクション・コール59H(拡張エラーの取得)を使用してください。

ファンクション・コール11Hを使って、一致するファイル名が見つけられた後、ファンクション・コール12Hはあいまいなリクエストに対して次の一致を見つけるためにコールされることがあります。

DTAは、前回の「最初のエントリーの探索」または「次のエントリー探索」からの情報を含みます。名前／エクステンション・フィールドを除くFCBのすべては、探索を続けるために必要な情報を、保持するために使用されます。したがって、このFCBが以前のファンクション・コール11Hまたは12Hと、このファンクション・コールの間にあれば、ディスク操作は実行されません。

13H —
ファイルの削除

目的

指定されたファイル名と一致するすべての現行ディレクトリーのファイル・エントリーを削除します。読み取り専用のファイルは指定できません。

例

```
MOV  AX,SEG FCB        ; FCBパラメーター・ブロックへアドレス
MOV  DS,AX
MOV  DX,OFFSET FCB
MOV  AH,13H            ; ファンクション・コール-FCBファイルの検索
INT  21H               ; DOSにリクエストを発行
CMP  AL,0              ; ファイルが削除された?
JNE  Error             ; エラーなし!

-----

FCB           LABEL   BYTE
Drive         DB       0                ; ドライブ
FName         DB      "Filename"        ; ファイル名(ブランクで埋まる、
                                        ; ?も使用可)
Ext           DB      "Ext"             ; ファイル・エクステンション(ブランク
                                        ; で埋まる、?も使用可)
              DB       25 DUP(0)        ; DOSにより埋められる
```

解説

ファイルが見つかった場合、ALは00Hです。

ファイルが見つからなかった場合、ALはFFHです。

実際のエラー状態を決定するためにはファンクション・コール59H(拡張エラーの取得)を使用してください。

すべての一致する現行ディレクトリー・エントリーが削除されます。グローバル・ファイル名文字“?”は、ファイル名またはエクステンションで使用できます。ディレクトリー・エントリーが一致しなければALはFFHを返し、一致する場合はALは00Hを返します。

ファイルが読み取り専用モードで指定されていると、ファイルは削除されません。

注: 削除する前にオープンしているファイルを閉じてください。

ネットワーク・アクセス権:　作成アクセス権が必要です。

14H —
順次読み取り

目的

現行ブロック(FCBバイトCH-0DH)とディスク転送アドレス(DTA)の現行レコード(FCBバイト1FH)によってアドレスされたレコードをロードし、その後レコード・アドレスを増加させます。

例

```
MOV  AX,SEG DTA          ; データ・バッファーへアドレス
MOV  DS,AX
MOV  DX,OFFSET DTA
MOV  AH,1AH              ; ファンクション・コール-DTAアドレスの設定
INT  21H                 ; DOSにリクエストを発行
MOV  AX,SEG FCB          ; FCBパラメータ・ブロックへアドレス
MOV  DS,AX
MOV  DX,OFFSET FCB
MOV  AH,14H              ; ファンクション・コール-FCB順次読み取り
INT  21H                 ; DOSにリクエストを発行
CMP  AL,0                ; データを読んだ?
JNE  Error               ; エラーなし!

-----

FCB           LABEL   BYTE
; すぐ前のオープンによって設定
DTA           LABEL   BYTE
              DB      ?Dup(0)   ;I/Oバッファー
```

解説

読み取りが成功した場合、ALは00Hです。

読み取りファイルがファイルの終了(EOF)の場合、ALは01Hです。

DTAがあまりに小さくて(読み取りが完了しなかったために)読み取りが循環する、またはオーバーフローした場合、ALは02Hです。

EOF(一部のレコードが読み込まれ、0で埋められる)ならば,ALは03Hになります。

実際のエラー状態を決定するためにはファンクション・コール59H(拡張エラーの取得)を使用してください。レコードの長さは、FCBレコード・サイズ・フィールドによって決定されます。

ネットワーク・アクセス権:　読み取りアクセス権が必要。

15H —
順次書き込み

目的

現行ブロックとディスク転送アドレスからのレコード・フィールド(FCBレコード・サイズ・フィールドによって決定されたサイズ)によってアドレスされたレコードを書き込みます。レコードがセクター・サイズより小さい場合、バッファーに書き込まれ、1セクター分のデータが溜まると、最終的に書き込みます。そしてレコード・アドレスが増やされます。

例

```
MOV  AX,SEG DTA          ; データ・バッファーへアドレス
MOV  DS,AX
MOV  DX,OFFSET DTA
MOV  AH,1AH              ; ファンクション・コール-DTAアドレスの設定
INT  21H                 ; DOSにリクエストを発行
MOV  AX,SEG FCB          ; FCBパラメータ・ブロックへアドレス
MOV  DS,AX
MOV  DX,OFFSET FCB
MOV  AH,15H              ; ファンクション・コール-FCB順次書き込み
INT  21H                 ; DOSにリクエストを発行
CMP  AL,0                ; データが書き込まれた?
JNE  Error               ; エラーなし!

FCB    LABEL   BYTE
; すぐ前のオープンによって設定
DTA    LABEL   BYTE
       DB      ?Dup(0)     ;I/Oバッファー
```

解説

書き込みが成功するとALは00Hです。

ディスクまたはディスケットがいっぱいの場合ALは01Hです(書き込みは、取り消されます)。

DTAがあまりに小さいために書き込みが、循環またはオーバーフローするとALは02Hです(書き込みは取り消されます)。

実際のエラー状態を決定するためにはファンクション・コール59H(拡張エラーの取得)を使用してください。ファイルが読み取り専用モードで指定された場合、順次書き込みは実行されず、01HがALで返されます。

ネットワーク・アクセス権:　書き込みアクセス権が必要。

16H —
ファイルの作成

目的

新しいファイルを作成します。

例

```
MOV  AX,SEG FCB    ; FCBパラメータ・ブロックへアドレス
MOV  DS,AX
MOV  DX,OFFSET FCB
MOV  AH,16H        ; ファンクション・コール-FCBファイルの作成
INT  21H           ; DOSにリクエストを発行
CMP  AL,0          ; ファイルは作成されオープンしているか?
JNE  Error         ; エラーなし!

 -----

FCB          LABEL   BYTE
Fdrive       DB      0            ; ドライブ (0=現行、1=A、2=B、...)
Fname        DB      "FILENAME"   ; ファイル名(ブランクで埋まる)
Fext         DB      "EXT"        ; ファイル・エクステンション(ブランク
                                  ; で埋まる)
             DB      25 DUP(0)    ; DOSにより埋められる
```

解説

ファイルが作成され、オープンされるとALは00Hです。

ファィルが作成されないとALはFFHです(通常、ディレクトリーがいっぱいか、ディスクがいっぱいの場合)。

実際のエラー状態を決定するためには、ファンクション・コール59H(拡張エラーの取得)を使用してください。

一致するファイル・エントリーが見つけられると、それが再使用されます。見つからなければ、ディレクトリーは空のファイル・エントリーとして探索されます。一致するものが見つかると、ファイル・エントリーは長さ０のファイルとして初期化され、ファイルがオープンされ(ファンクション・コール0FH参照）、ALに00Hが返されます。

ファイルは、適当な属性バイトを持つ拡張FCBを使用することによって、その作成の間に隠しファイルのマークを付けることもできます。

ネットワーク・アクセス権:　作成アクセス権が必要。

17H —
ファイル名の変更

目的

指定ドライブの現行ディレクトリーの第一ファイル名と一致する、すべてのファイル名を、２つのファイルが同じ名前とエクステンションを持たないという制限のもとで、第２のファイル名に変更します。

例

```
MOV  AX,SEG FCB       ; FCBパラメータ・ブロックへアドレス
MOV  DS,AX
MOV  DX,OFFSET FCB
MOV  AH,17H           ; ファンクション・コール-FCBファイル名の変更
INT  21H              ; DOSにリクエストを発行
CMP  AL,0             ; ファイル名は変更された?
JNE  Error            ; エラーなし!

 -----

 FCB      LABEL   BYTE
 Fdrive   DB      0           ; ドライブ(0=現行, 1=A, 2=B, ...)
 Fname    DB      "FNAME1"    ; ファイル名(ブランクで埋まる、?も使用可)
 Fext     DB      "EXT"       ; ファイル・エクステンション(ブランク
                              ; で埋まる、?も使用可)
          DB      5 DUP(0)    ; 予約済み
 NewName  DB      "FNAME2"    ; 新ファイル名(ブランクで埋まる、?も使用可)
 NewExt   DB      "EXT"       ; 新ファイル・エクステンション(ブランク
                              ; で埋まる、?も使用可)
          DB      7 DUP(0)    ; 予約済み
```

解説

１つまたは複数のファイル名が変更されるとALは00Hです。

現行ディレクトリーのファイルが一致しない、または新しい名前がすでに存在する場合、ALはFFHです。

実際のエラー状態を決定するためには、ファンクション・コール(拡張エラーの取得)を使用してください。

この変更されたFCBは、ドライブ・コードとファイル名を通常の位置に、そして第２のファイル名を第１のものの６バイト後(DS:DX+11H)の予約済み領域に持っています。第２のファイル名に“?”があると、元の名の対応する位置の文字は変更されません。

ファイルが読み取り専用モードで指定されていると、ファイル名は変更されません。

ネットワークアクセス権:　作成アクセス権が必要。

19H —
現行ディスク

目的

現行省略時ディスクを判別します。

例

```
        MOV     AH,19H   ; ファンクション・コール-現行ディスクの取得
        INT     21H      ; DOSにリクエストを発行
        MOV     Disk,AL  ; 現行ディスクを保存
  ----

  Disk   DB     ?        ; 現行ディスク・コード (0=A:, 1=B:, ...)
```

解説

現行省略時ドライブ(０=A、１=Bなど)のコードが、ALで返ります。

1AH —
ディスク転送アドレス(DTA)の設定

目的

DS:DXにディスク転送アドレスを設定します。

例

```
 MOV  AX,SEG DTA       ; バッファーへアドレス
 MOV  DS,AX
 MOV  DX,OFFSET DTA
 MOV  AH,1AH           ; ファンクション・コール-DTAアドレスの設定
 INT  21H              ; DOSにリクエストを発行
  -----

DTA    LABEL   BYTE    ; データ・バッファー
```

解説

このコールによって定義された域は、DS:DXのアドレスからDSのセグメントの最後までです。DOS/Vはセグメントの中でディスク転送が循環したり、次のセグメントにオーバーフローすることを許しません。DTAを設定しなければ、省略時DTAは、プログラム・セグメント・プリフィックスでオフセット80Hです。DTAを得るためにファンクション・コール2FHを発行します。

1BH —
割り振りテーブル情報

目的

省略時ドライブの割り振りテーブルについての情報を返します。

例

```
 MOV  AH,1BH                  ; ファンクション・コール-
                              ; 割り振りテーブル情報
 INT  21H                     ; DOSにリクエストを発行
 MOV  NumAllocUnits,DX        ; 割り振りユニットの数を保存
 MOV  NumSectsAllocUnit,AL    ; セクター／割り振りユニットの数を保存
 MOV  SectSize,CX             ; セクター・サイズの保存
 MOV  WORD PTR MediaType@+0,BX ; メディア・タイプ・バイトへの
                              ; ポインター
 MOV  WORD PTR MediaType@+2,DS

-----

NumAllocUnits     DW      ?        ; 現行ドライブの割り振りユニット数
NumSectsAllocUnit DB      ?        ; 割り振りユニット内のセクター数
SectSize          DW      ?        ; セクター・サイズ
MediaType@        DD      ?        ; メディア・タイプ・バイトへのポインター
```

解説

ファンクション・コール36H(ディスクの空き領域の取得)を参照してください。

1CH —
指定装置の割り振りテーブル情報

目的

指定装置の割り振りテーブル情報を返します。

例

```
 MOV   AH,1CH                  ; ファンクション・コール-
                               ; 割り振りテーブル情報
 MOV   DL,Drive                ; リクエストしたドライブ(0=現行、
                               ; 1=A:、...)
 INT   21H                     ; DOSにリクエストを発行
 MOV   NumAllocUnits,DX        ; 割り振りユニット数の保存
 MOV   NumSectsAllocUnit,AL    ; セクター／割り振りユニット数の保存
 MOV   SectSize,CX             ; セクター・サイズの保存
 MOV   WORD PTR MediaType@+0,BX ; メディア・タイプ・バイトの
                               ; ポインターの保存
 MOV   WORD PTR MediaType@+2,DS

-----
Drive             DB           ; 情報取得のためのドライブ番号
NumAllocUnits     DW    ?      ; 指定ドライブの割り振りユニット数
NumSectsAllocUnit DB    ?      ; 割り振りユニット中のセクター数
SectSize          DW    ?      ; セクター・サイズ
MediaType@        DD    ?      ; メディア・タイプ・バイトへのポインター
```

解説

このコールは、DLエントリーにドライブ番号など必要な情報(０=省略時、１=Aなど)を含むことを除いて、1BHのコールと同じです。DOS/Vディスク割り当てについてのより詳細の情報は、2-3ページの『ディスク・ディレクトリー』を参照してください。また、ファンクション・コール36H(ディスクの空き領域の取得)も参照してください。

21H —
ランダム読み取り

目的

現行ディスク転送アドレスのメモリーに、現行ブロックと現行レコード・フィールドによってアドレスされたレコードを読み取ります。

例

```
                                ; FCBの設定
 MOV  AX,SEG DTA                ; データ・バッファーへアドレス
 MOV  DS,AX
 MOV  DX,OFFSET DTA
 MOV  AH,1AH                    ; DTAアドレスの設定機能
 INT  21H                       ; DOSにリクエストを発行
 MOV  AX,SEG FCB                ; FCBパラメーター・ブロックへアドレス
 MOV  DS,AX
 MOV  DX,OFFSET FCB
 MOV  AH,21H                    ; ファンクション・コール-FCBランダム読み取り
 INT  21H                       ; DOSにリクエストを発行
 CMP  AL,0                      ; データを読んだ?
 JNE  Error                     ; エラーなし!

-----

FCB            LABEL   BYTE
; すぐ前のオープンにより設定
; DTAラベル・バイト
```

解説

読み取りが成功するとALは00Hです。

読み取りファイルがファイルの終了(EOF)の場合、ALは01Hです(データは読まない)。

DTAが小さくて(読み取りが完了しなかったため)読み取りが循環、またはオーバーフローした場合、ALは02Hです。

EOF(一部のレコードが読み取られ、0で埋められた)ならばALは03Hです。

実際のエラー状態を決定するためにはファンクション・コール59H(拡張エラーの取得)を使用してください。

現行ブロックと現行レコード・フィールドは、ランダム・レコード・フィールドと一致するように設定されます。これらのフィールドによってアドレスされたレコードは、現

行ディスク転送アドレスのメモリーに読み込まれます。レコード・サイズに関する情報は、4-1ページの第4章、『ファイル制御ブロックを使用したファイルへのアクセス』.を参照してください。

注: 機能24Hはこの機能を使う前にコールしなければなりません。

ネットワーク・アクセス権: 読み取りアクセス権が必要。

22H —
ランダム書き込み

目的

現行ディスク転送アドレス(DTA)から、現行ブロックと現行レコード・フィールドによってアドレスされたレコードを書き込みます。

例

```
                        ; FCBの設定
 MOV  AX,SEG FCB        ; FCBパラメーター・ブロックへアドレス
 MOV  DS,AX
 MOV  DX,OFFSET FCB
 MOV  AH,24H            ; ファンクション・コール-FCBの設定
                        ; 相対レコード・フィールド
 INT  21H               ; DOSにリクエストを発行
 MOV  AX,SEG DTA        ; データ・バッファーへアドレス
 MOV  DS,AX
 MOV  DX,OFFSET DTA
 MOV  AH,1AH            ; DTAアドレス設定機能
 INT  21H               ; DOSにリクエストを発行
 MOV  AX,SEG FCB        ; FCB パラメーター・ブロックへアドレス
 MOV  DS,AX
 MOV  DX,OFFSET FCB
 MOV  AH,22H            ; ファンクション・コール-FCBランダム読み取り
 INT  21H               ; DOSにリクエストを発行
 CMP  AL,0              ; データが書き込まれた?
 JNE  Error             ; エラーなし!

-----

FCB           LABEL   BYTE
; すぐ前のオープンにより設定
DTA           LABEL   BYTE
```

解説

書き込みが成功するとALは00Hです。

ディスクまたはディスケットがいっぱいの場合、ALは01Hです(書き込みは取り消されます)。

DTAが小さいために、書き込みが循環したり、オーバーフローした場合ALは02Hです(書き込みは取り消されます)。

実際のエラー状態を決定するためにはファンクション・コール59H(拡張エラーの取得)を使用してください。

現行ブロックと現行レコード・フィールドは、ランダム・レコード・フィールドと一致するように設定されます。それから、これらのフィールドによってアドレスされたレコードがディスク転送アドレスから(またはセクター・サイズと異なるレコードつまりバッファーされた場合)書き込まれます。

ファイルが読み取り専用モードで指定されていると、ランダム書き込みは実行されません。

ネットワーク・アクセス権:　書き込みアクセス権が必要。

23H —
ファイル・サイズ

目的

指定されたファイルと一致するエントリーを現行ディレクトリーで探索し、FCBランダム・レコード・フィールドにファイルのレコード数を設定します。

例

```
 MOV  AX,SEG FCB       ; FCB パラメータ・ブロックへアドレス
 MOV  DS,AX
 MOV  DX,OFFSET FCB
 MOV  AH,23H           ; ファンクション・コール-FCBファイル・サイズ
 INT  21H              ; DOSにリクエストを発行
 CMP  AL,0             ; ファイルは見つかった?
 JNE  Error            ; エラーなし!

-----

FCB           LABEL   BYTE
Drive         DB      0                ; ドライブ(0=現行, 1=A, 2=B, ...)
Name          DB      "FILENAME"       ; ファイル名(ブランクで埋まる)
Ext           DB      "EXT"            ; ファイル・エクステンション
                                       ; (ブランクで埋まる)
              DB      25 DUP(0)        ; DOSにより埋められる
```

解説

ファイルがある場合は、ALは00Hです。

ファイルが作成されていない場合は(通常、ディレクトリーがいっぱいか、ディスクがいっぱい)、ALはFFHです。

実際のエラー状態を決定するためにはファンクション・コール59H(拡張エラーの取得)を使用してください。

ディレクトリーは、一致するエントリーに対して探索されます。 一致するエントリーが見つかるとランダム・レコード・フィールドは、ファイルのレコード数に設定されます(レコード・サイズ・フィールド単位で切り上げ)。一致するファイルが見つからなければ、ALはFFHを返します。

注: この機能を使用する前にFCBレコード・サイズ・フィールドを設定しないと正しい情報が返されません。

24H —
相対レコード・フィールドの設定

目的

現行ブロックおよびレコード・フィールドと同じファイル・アドレスにランダム・レコード・フィールドを設定します。

例

```
MOV  AX,SEG FCB    ; FCB パラメータ・ブロックへのアドレス
MOV  DS,AX
MOV  DX,OFFSET FCB
MOV  AH,24H        ; ファンクション・コール-FCBの設定
                   ; 相対レコード・フィールド
INT  21H           ; DOSにリクエストを発行

-----

FCB           LABEL   BYTE
; すぐ前のオープンにより設定
```

解説

ランダム読み書きおよびランダム・ブロック読み書きを実行する前に、この機能をコールしなければなりません。

25H —
割り込みベクトルの設定

目的

指定した割り込み番号の割り込みベクトル・テーブルを設定します。

例

```
MOV  AX,SEG Handler    ; 新しいハンドラーへアドレス
MOV  DS,AX
MOV  DX,OFFSET Handler
MOV  AH,25H            ; ファンクション・コール-割り込みベクトルの設定
MOV  AL,Vector
INT  21H               ; DOSにリクエストを発行

-----

Vector     DB    ?     ; 置き換えられるベクトル数
Handler:               ; 割り込み処理コード
```

解説

ALで指定される割り込み番号のための割り込みベクトル・テーブルは、DS:DXに含まれるアドレスに設定されます。割り込みベクトルの内容を知るには、ファンクション・コール35H(割り込みベクトルの取得)を使用してください。

26H ―― 新しいプログラム・セグメントの作成

目的

新しいプログラム・セグメントを作成します。

例

```
 MOV     AH,26H               ; ファンクション・コール-プログラム・
                              ; セグメントの作成
 MOV     DX,SEG PSP           ; 新しいPSPを作成するセグメントのアドレス
 INT     21H                  ; DOSにリクエストを発行

-----

PSP           LABEL   BYTE    ; 埋められる領域
              DB      100H DUP(0)
```

解説

現行プログラム・セグメントの位置０からの100H領域全体が新しいプログラム・セグメントの位置０に複写されます。新しいセグメントの位置６のメモリー・サイズ情報は更新され、割り込み22H、23H、24Hの割り込みのベクトル・テーブル・エントリーから現行終了、Ctrl-Break出口、重大エラーの各アドレスが、0AHで始まる新しいプログラム・セグメントに保存されます。これらはプログラムが終了する時、この領域から復元されます。

注: EXECファンクション・コール4BHは、より完全なサービスを提供します。EXEC 4BHを使用してください。また、このコールを使用することは避けてください。

27H —
ランダム・ブロック読み取り

目的

指定されたレコード数(レコード・サイズ・フィールド単位)をランダム・レコード・フィールドで指定されたファイル・アドレスからディスク転送アドレスに読み取ります。

例

```
                                ; ディスク転送アドレスの設定
MOV  AX,SEG DTA                 ; データ・バッファーへアドレス
MOV  DS,AX
MOV  DX,OFFSET DTA
MOV  AH,1AH                     ; DTAアドレス設定機能
INT  21H                        ; DOSにリクエストを発行

MOV  AX,SEG FCB                 ; FCBパラメーター・ブロックへアドレス
MOV  DS,AX
MOV  DX,OFFSET FCB
MOV  AH,24H                     ; ファンクション・コール-FCBの設定
                                ; 相対レコード・フィールド
INT  21H                        ; DOSにリクエストを発行
MOV  AX,SEG FCB                 ; FCBパラメーター・ブロックへアドレス
MOV  DS,AX
MOV  DX,OFFSET FCB
MOV  CX,Records_to_read         ; 読み取るレコード数
MOV  AH,27H                     ; ファンクション・コール-FCBランダム・
                                ; ブロック読み取り
INT  21H                        ; DOSにリクエストを発行
CMP  AL,0                       ; データを読んだ?
JNE  Error                      ; エラーなし!

 -----

 FCB               LABEL   BYTE
 ; すぐ前のオープンにより設定
 DTA               LABEL   BYTE
                   DB      ?Dup(0)     ;I/Oバッファー
 Records_to_read   DW     ?
```

解説

読み取りに成功すればALは00Hです。

読み取りファイルがファイルの終了(EOF)の場合、ALは01HHです(データは読まれません)。

DTAが小さいために、読み取りが循環したり、オーバーフローした場合、ALは02Hです(読み取りは完了しません)。

EOF(一部のレコードが読み込まれ０で埋まる)の場合ALは03Hです。

実際のエラー状態を決定するためにはファンクション・コール59H(拡張エラーの取得)を使用してください。

ランダム・レコード・フィールドと現行ブロック／レコード・フィールドは、次のレコードのアドレスに設定されます(最初のレコードは読まれません)。

注:　ファンクション・コール24Hはこの機能より前にコールさなければなりません。

ネットワーク・アクセス権:　読み取りアクセス権が必要。

28H —
ランダム・ブロック書き込み

目的

ランダム・レコード・フィールドによって指定されたファイル・アドレスにディスク転送アドレスから指定されたレコード数を書き込みます。

例

```
                              ; ディスク転送アドレスの設定
MOV  AX,SEG DTA               ; データ・バッファーへアドレス
MOV  DS,AX
MOV  DX,OFFSET DTA
MOV  AH,1AH                   ; DTAアドレス設定機能
INT  21H                      ; DOSにリクエストを発行

MOV  AX,SEG FCB               ; FCBパラメーター・ブロックへアドレス
MOV  DS,AX
MOV  DX,OFFSET FCB
MOV  AH,24H                   ; ファンクション・コール-FCBの設定
                              ; 相対レコード・フィールド
INT  21H                      ; DOSにリクエストを発行


MOV  AX,SEG FCB               ; FCBパラメーター・ブロックへアドレス
MOV  DS,AX
MOV  DX,OFFSET FCB
MOV  CX,Records_to_write      ; 書き込むレコード数
MOV  AH,28H                   ; ファンクション・コール-FCBランダム・
                              ; ブロック書き込み
INT  21H                      ; DOSにリクエストを発行
CMP  AL,0                     ; データが書き込まれた?
JNE  Error                    ; エラーなし!

  -----

DTA     LABEL     BYTE

        DB        ?DUP(0)     ; I/Oバッファー
```

解説

書き込みが成功した場合、ALは00Hです。

ディスクまたはディスケットがいっぱいの場合、ALは01Hです(書き込みは取り消されます)。

DTAが小さくて、書き込みが循環したりオーバーフローした場合、ALは02Hです(書き込みは取り消されます)。

実際のエラー状態を決定するためにはファンクション・コール59H(拡張エラーの取得)を使用してください。

ディスクに十分な空間がない場合、ALは01Hを返し,レコードは書き込まれません。CXがエントリー時に０の場合レコードは書き込まれませんが、ファイルは現行ファイル・サイズより長くても短くても、ランダム・レコード・フィールドによって指定された長さに設定されます(割り振りユニットは解放されるか、適当に割り当てられます)。

注: このコールを使用する前にファンクション・コール24Hがコールしなければなりません。

ネットワーク・アクセス権: 書き込みアクセス権が必要。

29H —
ファイル名解析

目的

指定されたファイル名を解析します。

例

```
MOV   AX,SEG CmdBuf
MOV   DS,AX            ; コマンド文字列へアドレス
MOV   SI,OFFSET CmdBuf
MOV   AX,SEG FCB
MOV   ES,AX            ; FCBパラメータ・ブロックへアドレス
MOV   DI,OFFSET FCB
MOV   AH,29H           ; ファンクション・コール-FCBファイル名解析
MOV   AL,OPTIONS       ; 希望する動作を設定
INT   21H              ; DOSにリクエストを発行

CMP   AL,-1            ; ドライブは有効？
JE    Error            ; エラーなし！
-----

CmdBuf      LABEL    BYTE
            DB       " a:file.ext      ",0DH

FCB         LABEL    BYTE
; 見つかった入力を基にオープン前状態で作成される
  Options   DB   ?   ; 解析オプション
```

解説

ALの内容は下に示すような行う動作を決定するために使用されます。

<ビット７～４は０でなければならない>
bit: 7 6 5 4 3 2 1 0

ビット0=1の場合、先行する区切り記号は、DS:SIのコマンド行で走査され無視されます。そうでなければ、先行する区切り記号の走査後無視は行われません。

ビット1=1の場合、FCB内のドライブIDは、解析されるコマンド行でドライブが指定された場合のみ設定(変更)されます。

ビット2=1の場合、FCB内のファイル名は、コマンド行がファイル名を含む場合のみ変更されます。

ビット3=1の場合、FCB内のファイル名エクステンションは、コマンド行がファイル名エクステンションを含む場合のみ変更されます。

ファイル名内の区切り記号を以下に示します。: . ; , = +に加えてタブとスペースです。ファイル名終了記号はこれらの記号に加えて< > | / " [ ]および任意の制御文字です。

出力

グローバル文字(?または*)が「コマンド文字列」に見つからない場合、ALは00Hです。

グローバル文字(?または*)が「コマンド文字列」に見つかった場合、ALは01Hです。

指定されたドライブが無効な場合、ALはFFHです。

d:filename.extという形式のファイル名に対してコマンド行が解析され、もしそれが見つかれば、対応するオープンしていないFCBがES:DIに作成されます。ドライブ指定子がなければ、すべてブランクであるとみなされます。文字*がファイル名またはエクステンションに現れた場合、名前またはエクステンションの中の文字を含む、残りのすべての文字は?に設定されます。

DS:DIはファイル名の後の最初の文字を指して戻り、ES:DIはフォーマットされたFCBの最初のバイトを指します。有効なファイル名がない場合、ES:DI+1はひとつのブランクを含みます。

2AH —
日付の取得

目的

曜日、年、月、日を返します。

例

```
        MOV     AH,2AH          ; ファンクション・コール-日付の取得
        INT     21H             ; DOSにリクエストを発行
        MOV     DayofWeek,AL    ; 曜日の保存
        MOV     Year,CX         ; 年の保存
        MOV     Month,DH        ; 月の保存
        MOV     Day,DL          ; 日の保存

  -----

  DayofWeek     DB      ?       ; 0=日, ... 6=土
  Year          DW      ?       ; 1980から2099
  Month         DB      ?       ; １から12
  Day           DB      ?       ; １から31
```

解説

時計の一日の時刻が翌日にくり上がると、日付はそれに従って調整されます。各月の日数やうるう年も考慮されます。

2BH —
日付の設定

目的

日付を設定(CMOS時計があればそれも設定)します。

例

```
        MOV     AH,2BH          ; ファンクション・コール-日付の設定
        MOV     CX,Year         ; 年の設定
        MOV     DH,Month        ; 月の設定
        MOV     DL,Day          ; 日の設定
        INT     21H             ; DOSにリクエストを発行
        CMP     AL,0            ; 日付は有効か？
        JNE     Error           ; いいえ！

  -----

  Year          DW      ?       ; 1980から2099
  Month         DB      ?       ; １から12
  Day           DB      ?       ; １から31
```

解説

日付が有効で操作が成功すればALは00Hです。

日付が無効ならばALはFFHです。

エントリーでCX,DXはファンクション・コール2AHによって返されるのと同じ形式で、有効な日付を持っていなければなりません。

リターンの際は、日付が有効で設定操作が成功の場合、ALに00Hを返します。日付が無効の場合はALにFFHを返します。

2CH —
時刻の取得

目的

時、分、秒、100分の１秒が返されます。

例

```
        MOV     AH,2CH          ; ファンクション・コール-時刻の取得
        INT     21H             ; DOSにリクエストを発行
        MOV     Hour,CH         ; 時間の保存
        MOV     Minute,CL       ; 分の保存
        MOV     Second,DH       ; 秒の保存
        MOV     Hundredth,DL    ; 1/100秒の保存

  -----

  Hour          DB      ?       ; 0から23
  Minute        DB      ?       ; 0から59
  Second        DB      ?       ; 0から59
  Hundredth     DB      ?       ; 0から99
```

解説

エントリーでAHは2CHを含みます。リターンに際しては、CX,DXは１日の時刻を含みます。時刻は、実際には次のように４つの８ビット２進値で表されます。

CH 時間(０～23)
CL 分(０～59)
DH 秒(０～59)
DL 1/100秒(０～99)

この形式は、印刷できる形式に変換するのも容易ですし、またある時刻から別の時刻を減算するといった計算に用いることもできます。

2DH —
時刻の設定

目的

時刻を設定(CMOS時計があればそれも設定)します。

例

```
          MOV     AH,2DH          ; ファンクション・コール-時刻の設定
          MOV     CH,Hour         ; 時間の設定
          MOV     CL,Minute       ; 分の設定
          MOV     DH,Second       ; 秒の設定
          MOV     DL,Hundredth    ; 1/100秒の設定
          INT     21H             ; DOSにリクエストを発行
          CMP     AL,0            ; 有効な日付か？
          JNE     Error           ; いいえ！

    -----

    Hour          DB      ?       ; 0から23
    Minute        DB      ?       ; 0から59
    Second        DB      ?       ; 0から59
    Hundredth     DB      ?       ; 0から99
```

解説

時刻が有効ならばALは00Hです。

時刻が無効ならばALはFFHです。

エントリーでCX,DXは機能2CHによって返されるのと同じ形式で、時刻を持っています。リターンに際しては、時刻の要素のいずれかが無効の場合、設定操作が取り消され、ALはFFHを返します。

システムがCMOSのリアルタイム時計を持っている場合、それが設定されます。

2EH —
ベリファイ・スイッチの設定と解除

目的

ベリファイ(VERIFY: 検査)スイッチを設定または解除します。

例

```
; VERIFY=OFFに設定
    MOV     AH,2EH          ; ファンクション・コール-ベリファイの設定
    MOV     AL,0            ; オフに設定
    INT     21H             ; DOSにリクエストを発行
                            ; VERIFY=ONに設定
    MOV     AH,2EH          ; ファンクション・コール-ベリファイの設定
    MOV     AL,1            ; オンに設定
    INT     21H             ; DOSにリクエストを発行
```

解説

エントリーでALはベリファイ・スイッチをオンにするには01H、またオフにするには00Hでなければばなりません。ベリファイがオンの時、DOS/Vはデータ記録を確かなものにするために、ディスク書き込みを行うたびに、ベリファイ操作を行います。ディスク記録エラーは、非常にまれにしか起こりませんが、この機能は重要なデータの正しい書き込みを保証しなくてはならない、ユーザーのアプリケーション・プログラムのために提供されています。ファンクション・コール54Hを使用すると、ベリファイ・スイッチの現在値を知ることができます。

注: ベリファイ機能は、ネットワーク・ディスクに書き込まれたデータに対してはサポートされていません。

2FH —
ディスク転送アドレス(DTA)の取得

目的

現行ディスク転送アドレスを返します。

例

```
          MOV     AH,2FH              ; ファンクション・コール-
                                      ; ディスク転送アドレスの取得
          INT     21H                 ; DOSにリクエストを発行
          MOV     WORD PTR DTA@+0,BX  ; アドレス保存
          MOV     WORD PTR DTA@+2,ES

    -----

    DTA@          DD      ?           ; DTAバッファー
```

解説

エントリーでAHは2FHです。リターンの際は、ES:BXは現行ディスク転送アドレスを含みます。DTAはファンクション・コール1AHを使用して設定できます。

30H —
DOSバージョン番号の取得

目的

DOSのバージョン番号を返します。

例

```
        PUSH    CX              ; CXはコールで破壊される
        PUSH    BX
        MOV     AH,30H          ; ファンクション・コール-DOS/V
                                ; バージョンの取得
        INT     21H             ; DOSにリクエストを発行
        MOV     MajorVersion,AL ; バージョンの保存
        MOV     MinorVersion,AH
        MOV     DOS_Running_From,BH ;
        POP     BX
        POP     CX

  -----
  MajorVersion  DB      ?       ; X.YYのX
  MinorVersion  DB      ?       ; X.YYのYY
  DOS_Running_From  DB      ?       ; 0 = DOSはROMで動いていない
                                    ; 8 = DOSはROMで動いている
```

解説

エントリーでAHは30Hです。リターンの際は、CXは０に設定されます。ALはバージョン番号の整数部分です。AHはバージョン番号の小数部分です。DOSがROM内で動いているかいないかによって、BHの値は８または０です。

ALが０のバージョン番号の整数部分を返す場合、DOSバージョンはDOS 2.00以前とみなすことができます。ファンクション・コール33H AL=6（システム値の取得および設定）を使用して、バージョン番号を取得することができます。

31H —
常駐のままプロセス終了

目的

現行プロセスを終了し、初期割り振りブロックをメモリーにパラグラフ単位で設定を試みます。

例

```
        MOV     AH,31H          ; ファンクション・コール-プロセスの
                                ; 常駐終了
        MOV     AL,RetCode      ; ERRORLEVELの値の設定
        MOV     DX,MySize       ; プログラムとデータ・サイズの設定
        INT     21H             ; DOSにリクエストを発行

  -----

  RetCode       DB      ?       ; EXECに返す値
  MySize        DW      ?       ; コードとデータのサイズ
                                ; (パラグラフ単位)
```

解説

エントリーでALは２進数のリターン・コードを含みます。DXはパラグラフ単位のメモリーサイズ値を含みます。このファンクション・コールは、そのプロセスに属する他の割り振りブロックを解放しません。プロセスによってオープンされたファイルは、このコールが実行される時にクローズしません。ALで渡されるリターン・コードは、Wait(ファンクション・コール4DH)によりその親に受け渡しができ、バッチのERRORLEVELのサブコマンドでテストすることができます。

環境のコピーを含むブロックが終了前に割り振りが解除されれば、メモリーは有効に使用できます。これは、PSPの2Cに含まれるセグメントをESにロードし、ファンクション・コール49H(割り当てられたメモリー解放)を発行することによって行われます。５つの標準ハンドル0000から0004は終了前に閉じなければなりません。

33H —
システム値の取得および設定

目的

BREAK(Ctrl-Breakチェック)のような「システム値」の状態を設定または取得します。

例

```
; BREAK状態のチェック

        MOV     AH,33H          ; ファンクション・コール-システム値の
                                ; 取得および設定
        MOV     AL,0            ; BREAK取得を行う
        INT     21H             ; DOSにリクエストを発行
        MOV     BREAK,DL        ; 状態の保存 ストを発行

; BREAK=OFFの設定

        MOV     AH,33H          ; ファンクション・コール-システム値の
                                ; 取得および設定
        MOV     AL,1            ; BREAK取得を行う
        MOV     DL,0            ; OFFに設定
        INT     21H             ; DOSにリクエストを発行

; BREAK=ONの設定

        MOV     AH,33H          ; ファンクション・コール-システム値の
                                ; 取得および設定
        MOV     AL,1            ; BREAK設定を行う
        MOV     DL,1            ; ONの設定
        INT     21H             ; DOSにリクエストを発行

; ブート・ドライブの取得

        MOV     AH,33H          ; ファンクション・コール-システム値の
                                ; 取得および設定
        MOV     AL,5            ; ブート・ドライブの取得を行う
        INT     21H             ; DOSにリクエストを発行
        MOV     Drive,DL        ; ブート・ドライブの保存

; バージョン番号の取得

        MOV     AH,33H          ; ファンクション・コール-システム値の
                                ; 取得および設定
```

```
        MOV     AL,6            ; 真のバージョンの取得
        INT     21H             ; DOSにリクエストを発行
        MOV     MajorVersion,BL
        MOV     MinorVersion,BH
        MOV     Rev_Level,DL
        MOV     DOS_Flags,DH

  ----

  BREAK         DB      ?       ; 現行BREAKの状態(0=OFF, 1=ON)
  Drive         DB      ?       ; DOS/Vブート・ドライブ
  MajorVersion  DB      ?       ; 真のバージョン  X.YYのX
  MinorVersion  DB      ?       ; 真のバージョン  X.YYのYY
  Rev_Level   DB      ?       ;
  DOS_Flags   DB      ?       ;
```

解説

BREAKでは

Ctrl-Breakチェックの現在の状態をリクエストするには、ALを00Hにします。リターンの際は、DLは現在の状態(00H=OFF、01H=ON)になります。

状態を設定するにはALを01Hにします。DLは新しい状態(00H=OFF、01H=ON)になります。

ブート・ドライブでは

ブート・ドライブをリクエストするにはALを05Hにします。リターンとして、DLはドライブ番号(A:=1,C:=3,...)になります。

バージョンでは

DHのビット０から７は次の意味をもちます。

Bits 0 - 2 予約済み

Bit 3 １のときはDOSはROMで稼働中

Bit 4 １のときはDOSはHMAで稼働中

Bits 5 - 7 予約済み

35H —
割り込みベクトルの取得

目的

割り込みベクトルのアドレスを取得します。

例

```
        MOV   AH,35H                 ; ファンクション・コール-割り込み
                                     ; ベクトルの設定
        MOV   AL,Vector              ; 取得するベクトル(0から255)
        INT   21H                    ; DOSにリクエストを発行
        MOV   WORD PTR OldVect+0,BX
        MOV   WORD PTR OldVect+2,ES

  -----

  OldVect       DD      ?            ; 前回のベクトル内容
  Vector        DB      ?            ; 取得するベクトル番号
```

解説

エントリーでAHは35Hになります。ALは16進数の割り込み番号です。リターンの際は、ES:BXは指定された割り込みに対するCS:IP割り込みベクトルになります。割り込みベクトルの設定にはファンクション・コール25H(割り込みベクトル設定)を使用してください。

36H —
ディスク空き領域の取得

目的

ディスクの空き領域(利用できるクラスター、クラスター数／ドライブ、バイト数／セクター)を返します。

例

```
        MOV     AH,36H                  ; ファンクション・コール -
                                        ; ディスク空き領域の取得
        MOV     DL,Drive                ; 問い合わせるドライブ
                                        ; (0=現行, 1=A:,
                                        ; 2=B:, ...)
        INT     21H                     ; DOSにリクエストを発行
        CMP     AX,-1                   ; エラー?
        JE      Error                   ; はい
        MOV     SectAU,AX               ; 割り振りユニット・サイズの保存
        MOV     AvailAU,BX              ; 空き割り振りユニットの保存
        MOV     SectSize,CX             ; セクター・サイズの保存
        MOV     TotalAU,DX              ; ディスク・サイズの保存

        MOV     AX,SectSize             ; 割り振りユニット・サイズの計算
        MUL     SectAU
        MOV     CX,AX                   ; CX = バイト/AU
        MOV     AX,TotalAU              ; 総領域の計算
        MUL     CX
        MOV     WORD PTR TotalBytes+0,AX ; それを保存
        MOV     WORD PTR TotalBytes+2,DX
        MOV     AX,AvailAU              ; 空き領域の計算
        MUL     CX
        MOV     WORD PTR FreeBytes+0,AX ; それを保存
        MOV     WORD PTR FreeBytes+2,DX

  -----

  SectAU        DW      ?               ; 割り振りユニット内のセクター
  AvailAU       DW      ?               ; 空き割り振りユニット
  SectSize      DW      ?               ; セクター内のバイト
  TotalAU       DW      ?               ; DLディスク上の割り振り
                                        ; ユニットの数
  TotalBytes    DD      ?               ; ディスクのサイズ(バイト)
  FreeBytes     DD      ?               ; 空き領域(バイト)
  Drive         DD      ?               ; 情報を取得するドライブ番号
```

解説

DL内のドライブ番号が有効ならば、BXは利用できる割り振りユニットになり、DXはドライブ上の全割り振りユニット数になります。CXはセクターごとのバイト数になり、AXは各割り振りユニットに対するセクター数になります。

38H —
国別情報の取得または設定

目的

「現行の国コード」を設定するか、または国別情報を返します。

例

```
; 現在の国コードの設定

      MOV     AH,38H          ; ファンクション・コール-国別情報の取得
      MOV     AL,CountryID    ; 国ID(-1 if >= 255
      MOV     BX,CountryIDX   ; 国ID(if AL=-1)
      MOV     DX,-1           ; 国コード設定を示す
      INT     21H             ; DOSにリクエストを発行
      JC      Error           ; AX内にエラー・コード

; 国別情報の取得

      MOV     AX,SEG Buffer
      MOV     DS,AX
      MOV     DX,OFFSET Buffer
      MOV     AH,38H
      MOV     AL,CountryID    ; 国ID(-1 if >=255)
                              ; (0は現在の国コードの取得)
      MOV     BX,CountryIDX   ; 国ID(if AL=-1)
      INT     21H             ; DOSにリクエストを発行
      JC      Error           ; AX内にエラー・コード
      MOV     CountryCode,BX  ; 現在の国コードの取得

----

CountryCode   DW      ?       ; 現在の国コード
CountryIDX    DW      ?       ; 入力のための拡張国コード
Buffer        LABEL   WORD    ; 国情報(以下の形式参照)
CountryID     DB      ?       ; 入力のための国コード
```

国情報

```
DateFormat    DW    ?          ; データ形式：
                               ;  0 = m d y 順
                               ;  1 = d m y 順
                               ;  2 = y m d 順
$Symbol       DB    "????",0   ; 通貨記号
                               ;   例: "DM",0,?,?
Sep1000       DB    "?",0      ; ３桁ごとのセパレーター
                               ;   例: ",",0
Sep1          DB    "?",0      ; 少数セパレーター
                               ;   例: ".",0
SepDate       DB    "?",0      ; 日付セパレーター
                               ;   例: "/",0
SepTime       DB    "?",0      ; 時刻セパレーター
                               ;   例: ":",0
$Format       DB    ?          ; 通貨形式:
                               ;   0 = 通貨記号、値
                               ;   1 = 値、通貨記号
                               ;   2 = 通貨記号、空白、値
                               ;   3 = 値、空白、通貨記号
                               ;   4 = 通貨記号は10進セパレーター
SigDigits     DB    ?          ; 通貨の小数点以下の桁数
TimeFormat    DB    ?          ; 時刻形式
                               ;   0 = 12時間時計
                               ;   1 = 24時間時計
UpperCaseAL@  DD    ?          ; 大文字AL用のルーチン・アドレス
                               ; 値>=80Hのみ
SepData       DB    "?",0      ; データ・リスト・セパレーター
                               ;   例: ",",0
Reserved      DW    5 DUP(?)   ; 予約済み
```

解説

日付形式の意味は次のとおりです。

コード	日付
０=米国	m d y
１=ヨーロッパ	d m y
２=日本	y m d

大文字変換コール・アドレス:　大文字変換コールのためのレジスター内容は、次のとおりです。

エントリー	**レジスター内容**
AL	大文字に変換される文字のアスキー・コード

戻り	**レジスター内容**
AL	大文字入力文字のアスキー・コード

大文字変換コール・アドレスは形式が間接FARコール向きです。

戻り(リターン)

エラー・コードはAXに返されます。エラー・クラス、推奨される対処方法、位置についての追加情報のためには、ファンクション・コール59H(拡張エラーの取得)を発行してください。ファンクション・コール65H(拡張国別情報の取得)は、さらに詳しい国別情報を返すので、使用をお勧めします。

このファンクション・コールを使用して、「現在の国コード」を設定することはお勧めできません。ユーザーはCONFIG.SYSファイルにCOUNTRYコマンドを置くことで設定できます。設定された国別コードを変更しないでください。現在の国コードを変更するには、NLSFUNC DOS/Vエクステンションを導入する必要があります。

注:　この「国別情報の設定」およびファンクション・コール65H(拡張国別情報の取得)は**英語モード**のみで使用できます。

39H —
サブディレクトリーの作成(MKDIR)

目的

指定されたディレクトリーを作成します。

例

```
MOV  AX,SEG    DName  ; ディレクトリー名
MOV  DS,AX
MOV  DX,OFFSET DName
MOV  AH,39H           ; 機能-ディレクトリーの作成
INT  21H              ; DOSにリクエストを発行
JC   Error

 ----

DName          DB       "?? .. ??",0  ; ASCIIZ名
                                      ; 例：
                                      ; "c:¥dir",0
```

解説

エントリーで、DS:DXはドライブとディレクトリー・パス名を持つASCIIZ文字列のアドレスです。名前の最後以外のすべてのディレクトリー階層は、この機能を使用する前になければなりません。１度に１つのディレクトリー階層だけがこの機能で作成されます。ASCIIZ文字列の最大長さは64文字です。

エラー・コードはAXで返されます。エラー・レベル、推奨される対処方法、位置についての追加情報のためには、ファンクション・コール59H(拡張エラーの取得)を発行してください。

ネットワーク・アクセス権:　作成アクセス権が必要。

3AH —
サブディレクトリーの削除(RMDIR)

目的

指定されたディレクトリーを除去します。

例

```
MOV  AX,SEG DName              ; ディレクトリー名
MOV  DS,AX
MOV  DX,OFFSET DName
MOV  AH,3AH                    ; 機能-ディレクトリーの削除
INT  21H                       ; DOSにリクエストを発行
JC   Error

 ----

DName          DB      "?? .. ??",0  ; ASCIIZ名
                                     ;   例: "c:¥dir",0
```

解説

エントリーで、DS:DXはドライブとディレクトリー・パス名を持つASCIIZ文字列のアドレスです。指定されたディレクトリーはその構造から除去されます。現行ディレクトリーまたはその中にファイルを持つディレクトリーは除去されません。

エラー・コードはAXで返されます。エラー・クラス、推奨される対処方法、位置についての追加情報のためには、ファンクション・コール59H(拡張エラーの取得)を発行してください。

ネットワーク・アクセス権:　作成アクセス権が必要。

3BH —
現行ディレクトリーの変更(CHDIR)

目的

現行ディレクトリーを、指定されたディレクトリーに変更します。

例

```
MOV  AX,SEG DName              ; ディレクトリー名
MOV  DS,AX
MOV  DX,OFFSET DName
MOV  AH,3BH                    ; 機能-ディレクトリーの変更
INT  21H                       ; DOSにリクエストを発行
JC   Error

 ----

DName         DB       "?? .. ??",0  ; ASCIIZ名
                                     ;   例: "c:¥dir",0
```

解説

エントリーで、DS:DXはドライブとディレクトリー・パス名を持つASCIIZ文字列のアドレスです。文字列は64文字に制限され、ネットワーク・パスを含むことはできません。ディレクトリー・パスのメンバーのいずれかが存在しない場合、ディレクトリー・パスは変更されません。それ以外は、現行ディレクトリーがASCIIZ文字列に設定されます。

エラー・コードはAXで返されます。エラー・クラス、推奨される対処方法、位置についての追加情報のためには、ファンクション・コール59H(拡張エラーの取得)を発行してください。

3CH —
ファイルの作成(CREAT)

目的

新しいファイルを作成するか、または書き込みの準備として旧ファイルを長さを０に縮めます。

例

```
        MOV     AX,SEG FName        ;ファイル名
        MOV     DS,AX
        MOV     DX,OFFSET FName
        MOV     AH,3CH              ; 機能-ファイルの作成
        MOV     CX,Attribute        ; ファイルの属性
                                    ; 許される値
                                    ;  0001H=読み取り専用
                                    ;  0002H=隠しファイル
                                    ;  0004H=システム
                                    ;  0008H=ボリューム・ラベル
        INT     21H                 ; DOSにリクエストを発行
        JC      Error               ; AX内のエラー・コード
        MOV     Handle,AX           ; ファイル・ハンドルの保存

  ----

  FName         DB      "?? .. ??",0  ; ASCIIZ名
                                    ;   例: "c:¥dir¥file.ext",0
  Handle        DW      ?           ; ファイル・ハンドル
  Attribute     DW      ?           ; ディレクトリー・エントリーの属性
```

解説

ファイルが存在しなければ、ファイルが適当なディレクトリー内に作成され、ファイルは読み書き可能アクセス・コードを与えられます。ファイルは、読み書き可能状態でオープンされ、読み書きポインターがファイルの最初のバイトに設定されて、AXでハンドルが返されます。ファンクション・コール43H(ファイル・モードの変更)で、ファイルの属性を後で変えられることに注意してください。

エラー・コードはAXで返されます。エラー・クラス、推奨される対処方法、位置についての追加情報のためには、ファンクション・コール59H(拡張エラーの取得)を発行してください。

この機能は存在するボリューム・ラベルを置き換えません。このコールを発行する前に存在するボリューム・ラベルを削除しなければなりません。

ネットワーク・アクセス権:　作成アクセス権が必要。

3DH —
ファイルのオープン

目的

指定されたファイルをオープンします。

例

```
MOV  AX,SEG FName          ; ファイル名
MOV  DS,AX
MOV  DX,OFFSET FName
MOV  AH,3DH                ; 機能-ファイルのオープン
MOV  AL,OpenMode
INT  21H                   ; DOSにリクエストを発行
JC   Error                 ; AX内にエラー・コード
MOV  Handle,AX             ; 次の操作のためにファイル・ハンドルを保存

 ----

 FName         DB      "?? .. ??",0  ; ASCIIZ名
                                     ;   例: "c:¥dir¥file.ext",0
 Handle        DW      ?             ; ファイル・ハンドル
 OpenMode      DB      ?             ; オープン・モード
```

解説

読み書きポインターはファイルの最初のバイトで設定され、ファイルのレコード・サイズは1バイトです。読み書きポインターはファンクション・コール42Hで変更されます。返されたファイル・ハンドルは、引き続くファイルへの入出力に使用しなければなりません。ファイルの日付と時刻は、ファンクション・コール57Hで取得または設定ができます。そしてその属性は、ファンクション・コール43Hで読み出すことができます。

エラー・コードはAXで返されます。エラー・クラス、推奨される対処方法、位置についての追加情報にはファンクション・コール59H(拡張エラーの取得)を発行してください。

注:

1. このコールは、指定された名前と一致する任意の通常ファイルまたは隠しファイルをオープンします。

2. 装置名はコロンで終わることはできません。

3. ファイルがクローズされると、オープンによりそれに課せられた、どの共用制限も取り消されます。

4. ファイル共用がロードされているか、またはファイルが共用モードのネットワーク・ファイルでなければなりません。SHAREコマンドを参照してください。
5. ファイルの読み取り専用属性は、拡張FCBを使用してファイルを作成するか、CXで適当な属性を指定し、CHMODの割り込み21Hファンクション・コールを使用するか、DOS/V ATTRIBコマンドを使用して設定できます。
6. ファイルが従属プロセスに継承される場合、すべての共用とアクセス制限も引き継がれます。
7. オープン・ファイル・ハンドルが、DUPファンクション・コールのどちらかによって複製される場合、すべての共用とアクセス制限も複製されます。

オープン・モード

オープン・モードはALで定義され、４ビット指向フィールドで構成されます。

継承フラグ　オープンされたファイルが従属プロセスによって継承される場合に指定します。

共用モード・フィールド　他のプロセスがそのファイルに対して行える操作を定義します。

予約フィールド

アクセス・フィールド　現行プロセスがそのファイルに対して行える操作を定義します。

ビット・フィールド

ビット・フィールドは次のようにマップされます。

```
                         <I> < S > <R> < A >
オープン・モード・ビット   7   6 5 4  3   2 1 0
```

I　継承フラグ

I=０の場合、ファイルは従属プロセスによって継承されます。
I=１の場合、ファイルは現行プロセスに対して固有です。

S　共用モード

ファイルは次のようにオープンされます。

S = 000 — 互換モード
S = 001 — 読み書き拒否モード(排他的)
S = 010 — 書き込み拒否モード
S = 011 — 読み取り拒否モード
S = 100 — 非拒否モード

他のいずれの組み合わせも無効です。

ファイルをオープンする時、共用モードで他のプロセスがそのファイルに行える操作を、DOS/Vに知らせなければなりません。省略時の互換モードでは、ネットワークの他のすべてのコンピューターがそのファイルにアクセスするこ

とを拒否します。ユーザーのプロセスがそのファイルを操作している時に、他のプロセスがそれを読み続けることが可能な場合は、「書き込み拒否」を指定してください。「書き込み拒否」は他のプロセスによる書き込みは禁止しますが、読み取りは許可します。

同様に、ユーザーはプロセスが行うことのできる操作、またはそのアクセス・モードを指定しなければなりません。省略時のアクセス・モード、「読み書き可能」モードは、コンピューターの他のプロセスまたはネットワークの他のコンピューターが「非拒否」モード以外の共用モードでファイルをオープンしている場合、オープン・リクエストをfail(失敗)させます。そのファイルを読み取るだけならば、他のすべてのプロセスが「非拒否」または「書き込み拒否」モードのどちらかを指定すれば、オープンは成功します。ファイル共用は、２つの共用プロセスの協力を必要とします。

R 予約済み(このビット・フィールドは０に設定)

A アクセス

ファイル・アクセスは、次のように割り当てられます。

A=000　読み取りアクセス
A=001　書き込みアクセス
A=010　読み書きアクセス

他のいずれの組み合わせも無効です。

ネットワーク・アクセス権:　オープン・モード・フィールド(AL)のアクセス・フィールド(A)が以下と等しい場合

000 読み取りアクセス権が必要

001 書き込みアクセス権が必要

010 読み書きアクセス権が必要

互換モード

ファイルは次のファンクションコールでオープンされる場合、「互換モード」であると考えられます。

- 任意のCREATEファンクション・コール
- FCBファンクション・コール
- 指定された互換モードを持つハンドル・ファンクション・コール

ファイルは互換モードの単一プロセスで何度でもオープンすることができます。ただしファイルは他の４つの共用モードのうちの１つで、現在オープンされていないことが前提です。ファイルが読取り専用にマークされ、「読み取りアクセス」の「書き込み拒否」共用モードでオープンされている場合、ファイルは「読み取りアクセス」の「互換モード」でオープンされます。ファイルが他の共用モードの１つでオープンに成功し、そのファイルを「互換モード」で再度オープンしようとすると、このエラーを知らせる

ために割り込み24Hを生成します。基本割込み24Hエラーは**ドライブの準備ができていない**ことを示し、拡張エラーは**共用違反**を示します。

共用モード

互換モードでオープンされたファイルの共用モードは、ファイルの読み取り専用属性によってDOS/Vが変更します。これは、読み取り専用ファイルの共用を可能にします。

ファイル・オープンの手段	読み取り専用アクセス	共用モード
FCB	読取り専用	書き込み拒否
ハンドル読み取り	読取り専用	書き込み拒否
ハンドル書き込み	エラー	-----
ハンドル読み書き	エラー	-----

ファイル・オープンの手段	読み取り専用以外のアクセス	共用モード
FCB	読み書き	互換
ハンドル読み取り	読み取り	互換
ハンドル書き込み	書き込み	互換
ハンドル読み書き	読み書き	互換

読み書き拒否モード(排他的)

ファイルが「読み書き拒否」モードでのオープンに成功すると、ファイルへのアクセスは排他的です。このモードで現在オープンさているファイルは、ファイルがクローズされるまで、どのプロセス(現行プロセスを含む)による、どの共用モードでも再度オープンすることはできません。

書き込み拒否モード

「書き込み拒否」モードでオープンが成功したファイルは、ファイルがクローズするまでそのファイルに対する他の書き込みアクセス・オープン(A=001または010)を妨ぎます。「書き込み拒否」モードでファイルをオープンしようとする試みは、ファイルが書き込みアクセスでオープンしている場合には不成功に終わります。

読み取り拒否モード

「読み取り拒否」共用モードでオープンが成功したファイルは、ファイルが閉じられるまで、そのファイルに対する他の書き込み共用アクセス・オープン(A=001または010)を妨ぎます。「読み取り拒否」共用モードでファイルをオープンしようとする試みは、ファイルが互換性モードまたは読み取りアクセスでオープンしている場合は、不成功に終わります。

非拒否モード

「非拒否」モードでオープンに成功したファイルは、ファイルの読み書きアクセスに制限はありません。「非拒否」モードでファイルをオープンしようとする試みは、ファイルが「互換」モードでオープンされている場合には不成功に終わります。

ネットワーク・ディスクに常駐するファイルにアクセスする時、ファイルが次に示す共用モードのいずれかでオープンされている場合には、ローカル・バッファリングはされません。

- 読み取り拒否
- 非拒否

そのためネットワーク環境では、「読み書き拒否」共用モード、「互換」共用モードおよび「書き込み拒否」モード・オープンではローカル・バッファリングします。

次の共用マトリックスは、オープンの結果を示します。また同じファイルをすべてのアクセスと共用モードの組み合わせを使い、再オープンを試みた場合の結果を示します。

		DRW			DW			DR			ALL		
		I	IO	O	I	IO	O	I	IO	O	I	IO	O
DRW	I	N	N	N	N	N	N	N	N	N	N	N	N
	IO	N	N	N	N	N	N	N	N	N	N	N	N
	O	N	N	N	N	N	N	N	N	N	N	N	N
DW	I	N	N	N	Y	N	N	N	N	N	Y	N	N
	IO	N	N	N	N	N	N	N	N	N	Y	N	N
	O	N	N	N	N	N	N	Y	N	N	Y	N	N
DR	I	N	N	N	N	N	N	N	N	N	N	N	Y
	IO	N	N	N	N	N	N	N	N	N	N	N	Y
	O	N	N	N	N	N	N	N	N	Y	N	N	Y
ALL	I	N	N	N	Y	Y	Y	N	N	N	Y	Y	Y
	IO	N	N	N	N	N	N	N	N	N	Y	Y	Y
	O	N	N	N	N	N	N	Y	Y	Y	Y	Y	Y

Y　　:２回目、３回目、・・・のオープンが許される
N　　:２回目、３回目、・・・のオープンが許されない
DRW　:読み書き拒否モード(排他的)
DW　 :書き込み拒否モード
DR　 :読み取り拒否モード
ALL　:読み書きモード
I　　:読み取り専用アクセス
O　　:書き込み専用アクセス
IO　 :読み書きアクセス

3EH —
ファイル・ハンドルのクローズ

目的

指定されたファイル・ハンドルをクローズします。

例

```
          MOV     AH,3EH          ; ファンクション・コール-ハンドルの
                                  ; クローズ
          MOV     BX,Handle
          INT     21H             ; DOSにリクエストを発行
          JC      Error           ; AX内にエラー・コード

    ----

    Handle        DW      ?       ; ファイル・ハンドル(OpenまたはCreateから)
```

解説

エントリーで、BXはOpenまたはCreateによって返されたファイル・ハンドルです。リターンの際はファイルがクローズされ、ディレクトリーは更新されます。そして、そのファイルのためのすべての内部バッファーは空に(フラッシュ)されます。

エラー・コードはAXで返されます。エラー・クラス、推奨される対処方法、位置についての追加情報のためには、ファンクション・コール59H(拡張エラーの取得)を発行してください。

3FH —
ファイルまたは装置からの読み取り

目的

ファイルからバッファー位置に指定されたバイト数を転送します。

例

```
MOV  AX,SEG Buffer           ; データ・バッファーをアドレス
MOV  DS,AX
MOV  DX,OFFSET Buffer
MOV  AH,3FH                  ; 機能-ファイルからの読み取り
MOV  BX,Handle
MOV  CX,BufSize              ; バッファー・サイズ
INT  21H                     ; DOSにリクエストを発行
JC   Error                   ; AX内にエラー・コード
CMP  AX,0                    ; ファイルの終了(EOF)か？
JE   EOF                     ; はい！
MOV  SizeRead,AX             ; 読み取り合計の保存

 ----
 N             EQU   512      ; 典型的なバッファー・サイズ
 Handle        DW    ?        ; ファイル・ハンドル
                              ; (Openまたは作成Createから)
 BufSize       DW    N        ; バッファー・サイズ、Nは
 Buffer        DB    N DUP(?) ; データ・バッファー
 SizeRead      DW    ?        ; バッファー内データの合計
```

解説

エントリーで、BXはファイル・ハンドルを含みます。CXは読み込まれるバイト数です。DS:DXはバッファー・アドレスです。リターンの際は、AXは読み込まれたバイト数を含みます。

このファンクション・コールは、ファイルからバッファー位置へ(CX)バイトを転送しようとします。すべてのバイトが読み取られる保証はありません。たとえば、DOS/Vがキーボードから読み取る時、多くてテキスト１行が転送されるだけです。この読み取りが標準入力装置から行われれば、入力はリダイレクトできます。AXの値が０であれば、その時にはプログラムがファイルの終わりから読み取ろうとしたことを示しています。

エラー・コードはAXで返されます。エラー・クラス、推奨される対処方法、位置についての追加情報のためには、ファンクション・コール59H(拡張エラーの取得)を発行してください。

ネットワーク・アクセス権:　読み込みアクセス権が必要。

40H —
ファイルまたは装置への書き込み

目的

バッファーから指定されたファイルへ指定されたバイト数を転送します。

例

```
MOV  AX,SEG Buffer       ; データ・バッファー
MOV  DS,AX
MOV  DX,OFFSET Buffer
MOV  CX,BufSize
MOV  AH,40H              ; 機能-ファイルへの書き込み
MOV  BX,Handle
MOV  DX,OFFSET Buffer
INT  21H                 ; DOSにリクエストを発行
JC   Error               ; AX内にエラー・コード
CMP  AX,CX               ; ディスク空間が足りない？
JB   FullDisk            ; はい！

 ----
 N             EQU    512       ; 典型的なバッファー・サイズ
 Handle        DW     ?         ; ファイル・ハンドル
                                ; (OpenまたはCreateから)
 BufSize       DW     N         ; バッファー・サイズ
 Buffer        DB     N DUP(?)  ; データ・バッファー
```

解説

エントリーで、BXはファイル・ハンドルです。CXは書き込むバイト数です。DS:DXは書き込むデータのアドレスです。

このファンクション・コールは、バッファーからファイルへ(CX)バイトを転送しようとします。AXは、実際に書き込まれたバイト数を返します。桁上げ(キャリー)フラグが設定されず、この値が(CX)でリクエストされた数と同じでない場合、エラーと考えなければなりません。エラー・コードは返されませんが、ユーザーはプログラムでこれらの値を比較できます。通常、エラーの原因はディスク空間が足りないことです。この書き込みが標準出力装置に対して行われると、出力はリダイレクトできます。

ファイル・ポインターの現在の位置でファィルを打ち切るには、割り込み21Hを発行する前に(CX)のバイト数を０に設定してください。ファイル・ポインターは、読み書きおよびファンクション・コール42H(ファイル読み書きポインターの移動)の実行によって希望する位置に移動できます。

ファイルが読み取り専用の場合は、ファイルまたは装置への書き込みは実行されません。

エラー・コードはAXで返されます。エラー・クラス、推奨される対処方法、位置についての追加情報のためには、ファンクション・コール59H(拡張エラーの取得)を発行してください。

ネットワーク・アクセス権: 書き込みアクセス権が必要。

41H —
指定ディレクトリーからのファイルの削除(UNLINK)

目的

ファイル名と関連したディレクトリー・エントリーを除去します。

例

```
            MOV  AX,SEG FName       ; ファイル名
            MOV  DS,AX
            MOV  DX,OFFSET FName
            MOV  AH,41H             ; 機能-ファイルの削除
            INT  21H                ; DOSにリクエストを発行
            JC   Error              ; AX内にエラー・コード

  ----

 FName        DB    "?? .. ??",0  ; ASCIIZ名
                                  ;   例: "c:¥dir¥File.ext",0
```

解説

?や*のグローバル・ファイル名文字は、ASCIIZ文字列のどの部分にも許可されません。読み取り専用ファイルは、このコールでは削除できません。読み取り専用ファイルを削除するには、最初にファンクション・コール43Hを使い、ファイルの読み取り専用属性を0に変更してからファイルを削除します。

エラー・コードはAXで返されます。エラー・クラス、推奨される対処方法、位置についての追加情報のためには、ファンクション・コール59H(拡張エラーの取得)を発行してください。

ネットワーク・アクセス権:　作成アクセス権が必要。

42H —
ファイル読み書きポインターの移動(LSEEK)

目的

指定の方法に従って読み書きポインターを移動します。

例

```
        MOV     AH,42H                  ; ファンクション・コール-
                                        ; 読み書きポインターの移動
        MOV     AL,Method               ; 位置決めの方式：
                                        ;   0 = ファイルの最初(BOF)から
                                        ;   1 = 現在の位置から
                                        ;   2 = ファイルの終了(EOF)から
        MOV     BX,Handle               ; ファイルの選択
        MOV     DX,WORD PTR Position+0  ; 新しい位置 = 位置 + 方式
        MOV     CX,WORD PTR Position+2
        INT     21H                     ; DOSにリクエストを発行
        JC      Error                   ; AX内にエラー・コード
        MOV     WORD PTR Position+0,AX  ; 新しいファイル位置の設定
        MOV     WORD PTR Position+2,DX

  ----

  Handle        DW      ?               ; ファイル・ハンドル
                                        ; (OpenまたはCreateから)
  Position      DD      ?               ; ファイル・オフセット(おそらく負)
  Method        DB      ?
```

解説

エントリーで、ALは方式の値を含みます。BXはファイル・ハンドルです。CX:DXはバイトで希望するオフセットです。CXは最上位部です。リターンの際は、DX:AXはポインターの新しい位置になりキャリー・フラグが設定されていなければDXは最上位部を含みます。

エラー・コードはAXで返されます。エラー・クラス、推奨される対処方法、位置についての追加情報のためには、ファンクション・コール59H(拡張エラーの取得)を発行してください。

このファンクション・コールは、次の方式に従って読み書きポインターを移動します。

AL	意味
0	ポインターはファイルの始めからCX:DXバイト(オフセット)移動されます。
1	ポインターは現在の場所にオフセットを加えた位置に移動されます。
2	ポインターはファイルの終わり(EOF)にオフセットを加えた位置に移動されます。この方式はファイルの大きさを決定するために使用されます。

注: LSEEK操作が、「読み取り拒否」または「非拒否」共用モードのいずれかでオープンされている、ネットワーク・ディスク上に常駐するファイルに対して行われる場合、読み書きポインター情報は、ファイルが実際に存在するコンピューター上で調整されます。ファイルが他の共用モードでオープンされている場合、読み書きポインター情報は、リモート・コンピューターに保管されます。

43H —
ファイル・モードの変更(CHMOD)

目的

指定のファイルのファイル・モードを変更します。

例

```
; 属性の取得

      MOV     AX,SEG FName          ; ファイル名
      MOV     DS,AX
      MOV     DX,OFFSET FName       ; DS:DXはASCIIZパス名を指す
      MOV     AL,0                  ; 取得を示す
      MOV     AH,43H                ; 機能-ファイル・モードの変更
      INT     21H                   ; DOSにリクエストを発行
      JC      Error                 ; AX内にエラー・コード
      MOV     Attribute,CX          ; 属性の保存

; 属性の設定

      MOV    AX,SEG FName           ; ファイル名
      MOV    DS,AX
      MOV    DX,OFFSET FName        ; DS:DXはASCIIZパス名を指す
      MOV    AL,1                   ; 設定を示す
      MOV    AH,43H                 ; 機能-ファイル・モードの変更
      MOV    CX,Attribute           ; 属性の設定
      INT    21H                    ; DOSにリクエストを発行
      JC     Error                  ; AX内にエラー・コード

----

Fname         DB     64 Dup (0)    ;ASCIIZ名
                                   ;  例: "c:¥dir¥File.ext",0
Attribute     DW     ?             ;ファイルの属性
                                   ;  例: 0001Hは読み取り専用
```

解説

エントリーで、ALはファンクション・コードです。DS:DXはドライブ、パス、ファイル名を持つASCIIZ文字列のアドレスです。

ALが01Hの場合、ファイル属性はCX内の属性に設定されます。属性バイトについては、2-3ページの『ディスク・ディレクトリー』を参照してください。ALが00Hの場合、ファイルの現在の属性がCXで返されます。

エラー・コードはAXで返されます。エラー・クラス、推奨される対処方法、位置についての追加情報のためには、ファンクション・コール59H(拡張エラーの取得)を発行してください。

注: 保存(20H)、読み取り専用(01H)、システム(04H)、隠しファイル(02H)ビットのみが変更できます。CXの他のビットはすべて0でなければなりません。そうでなければ、エラーが表示されることがあります。

ネットワーク・アクセス権: 保存ビット(AL=20H)を変更するためにアクセス権は必要とされません。他のいずれかのビットを変更するために、作成アクセス権が必要です。

44H —
装置の入出力制御

目的

オープン装置ハンドルと関連する装置情報を設定または取得します。または、装置ハンドルへ制御文字列を送るか、装置ハンドルから制御文字列を受け取ります。

このファンクション・コールの詳細については、C-1ページの付録C、『装置の入出力制御(IOCtl)』を参照してください。

45H —
ファイル・ハンドルの複製(DUP)

目的

オープ・ファイルと、同じファイルの同じ位置を参照する、新しいファイル・ハンドルを返します。

例

```
        MOV     AH,45H          ; ファンクション・コール -
                                ; ファイル・ハンドルの複製
        MOV     BX,Handle       ; ファイルの選択
        INT     21H             ; DOSにリクエストを発行
        JC      Error           ; AX内にエラー・コード
        MOV     NewHandle,AX    ; 新しいハンドルを保存

  ----

  Handle        DW      ?       ; ファイル・ハンドル
                                ; (OpenまたはCreateから)
  NewHandle     DW      ?       ; 複製ハンドルのファイル・ハンドル
```

解説

エントリーで、BXはファイル・ハンドルです。リターンの際は、AXは返されたファイル・ハンドルになります。

エラー・コードはAXで返されます。エラー・クラス、推奨される対処方法、位置についての追加情報のためには、ファンクション・コール59H(拡張エラーの取得)を発行してください。

注: 読み書きポインターのどちらかのハンドルを、読み取り、書き込み、またはLSEEKファンクション・コールのいずれかで移動した場合、もう一方のハンドルのポインターも変更されます。

46H —
ファイル・ハンドルの強制複製(FORCDUP)

目的

CXのハンドルが、BX内のハンドルと同じファイルの同じ位置を示すように強制します。

例

```
        MOV     AH,46H          ; ファンクション・コール-
                                ; ハンドルの強制複製
        MOV     BX,Handle       ; ファイル選択
        MOV     CX,NewHandle    ; 新しいファイル定義の選択
        INT     21H             ; DOSにリクエストを発行
        JC      Error           ; AX内にエラー・コード

  ----

  Handle        DW      ?       ; ファイル・ハンドル
                                ; (OpenまたはCreateから)
  NewHandle     DW      ?       ; 複製されたファイル・ハンドル
```

解説

エントリーで、BXはファイル・ハンドルです。CXは次のファイル・ハンドルです。リターンの際は、CXファイル・ハンドルは、BXファイル・ハンドルと同じファイルの同じ位置を示します。CXファイル・ハンドルがオープンしたファイルだった場合、最初にそれはクローズします。どちらかのハンドルに読み書きポインターを移動した場合、他方のポインターも変更されます。

エラー・コードはAXで返されます。エラー・クラス、推奨される対処方法、位置についての追加情報のためには、ファンクション・コール59H(拡張エラーの取得)を発行してください。

47H —
現行ディレクトリーの取得

目的

指定されたドライブの現行ディレクトリーの(ルート・ディレクトリーから始まる)全パス名を、DS:SIによって示される領域に置きます。

例

```
 MOV     AX,SEG DName     ; ディレクトリー名バッファー
 MOV     DS,AX
 MOV     SI,OFFSET DName  ; DS:SIはバッファーを指す
 MOV     DL,Drive         ; ドライブ選択
 MOV     AH,47H           ; 機能-現行ディレクトリーの取得
 INT     21H              ; DOSにリクエストを発行
 JC      Error            ; AX内にエラー・コード

----

Drive    DB  ?            ; ドライブ(0=現行, 1=A:, 2=b:, ...)
DName    DB  64 DUP(?)    ; 返されたASCIIZディレクトリー名
                          ;   example: "dir1¥dir2",0
```

解説

ドライブ文字は返される文字列には含まれません。文字列は円記号(¥)では始まりません。また00Hを含むバイトで終わります。

エラー・コードはAXで返されます。エラー・クラス、推奨される対処方法、位置についての追加情報のためには、ファンクション・コール59H(拡張エラーの取得)を発行してください。

48H —
メモリーの割り当て

メモリーにリクエストされたパラグラフ数を割り当てます。

例

```
          MOV     AH,48H           ; ファンクション・コール-
                                   ; メモリーの割り当て
          MOV     BX,Paragraphs    ; リクエストしたパラグラフ
          INT     21H              ; DOSにリクエストを発行
          JNC     Done
          MOV     AH,48H           ; ファンクション・コール-
                                   ; メモリーの割り当て
                                   ; BXは最大可能メモリーを設定
          INT     21H              ; DOSにリクエストを発行
    Done:
          MOV     BlockSeg,AX      ; メモリーのBlockSegを保存
          MOV     Paragraphs,BX

    ----

    Paragraphs    DW      ?        ; パラグラフ単位でリクエストされたサイズ
                                   ; (割り当てられたバイトは16×パラグラフ)
    BlockSeg      DW      ?        ; 割り当てられたBlockingSegアドレス
```

解説

エントリーで、BXはリクエストされたパラグラフ数です。リターンの際は、AX:0は割り当てられたメモリー・ブロックを指します。割り当てが失敗すると、BXは利用できる最大ブロック・サイズをパラグラフ単位で返します。

エラー・コードはAXで返されます。エラー・クラス、推奨される対処方法、位置についての追加情報のためには、ファンクション・コール59H(拡張エラーの取得)を発行してください。

49H —
割り当てられたメモリーの解放

目的

指定された割り当てられたメモリーを解放します。

例

```
        MOV     AH,49H          ; ファンクション・コール-メモリーの解放
        MOV     ES,BlockSeg     ; 解放アドレスの設定
        INT     21H             ; DOSにリクエストを発行
        JC      Error

  ----

  BlockSeg       DW      ?      ; 割り当てられたメモリーの
                                ; BlockSegアドレス
```

解説

エントリーで、ESはシステム・プールに返されるブロックのセグメントです。リターンの際は、メモリーのブロックはシステム・プールに返されます。

エラー・コードはAXで返されます。エラー・クラス、推奨される対処方法、位置についての追加情報のためには、ファンクション・コール59H(拡張エラーの取得)を発行してください。

4AH —
割り当てられたメモリー・ブロックの変更(SETBLOCK)

目的

新しく指定されたブロック・サイズを含むために、割り当てられたメモリー・ブロックを変更します。

例

```
          MOV     AH,4AH          ; ファンクション・コール-
                                  ; 割り当てられたメモリーの変更
          MOV     ES,BlockSeg     ; 解放アドレスの設定
          MOV     BX,BlockSize    ; 新しいサイズ
          INT     21H             ; DOSにリクエストを発行
          JNC     Done
          MOV     AH,4AH          ; ファンクション・コール -
                                  ; メモリーの割り当て
                                  ; BXは最大可能メモリーを設定
          INT     21H             ; DOSにリクエストを発行
    Done:
          MOV     Size,BX

    ----

    BlockSeg      DW      ?       ; 割り当てられたメモリーの
                                  ; セグメント・アドレス
    BlockSize     DW      ?       ; パラグラフ単位でリクエストされたサイズ
                                  ; (割り当てられたバイトは16×サイズ)
```

解説

エラー・コードはAXで返されます。エラー・クラス、推奨される対処方法、位置についての追加情報のためには、ファンクション・コール59H(拡張エラーの取得)を発行してください。

注: このコールは、ファンクション・コール31H(常駐のままプロセス終了)を使用する前に、プログラムのサイズを設定するためによく使われます。プログラム・セグメント・プリフィックスを使用してください。この値は、ファンクション・コール62H(プログラム・セグメント・プリフィックス・アドレスの取得)を使用することで得られます。他の使い方は、ファンクション・コール4BH (プログラムのロードまたは実行)を使用する準備として、メモリーを解放することです。

4BH —
プログラムのロードまたは実行(EXEC)

目的

ひとつのプログラムが他のプログラムをメモリーにロードすることを可能にします。またその実行の開始も選択できます。

例

```
; プログラムの実行

 MOV  AH,4BH               ; ファンクション・コール-
                           ; プログラムの実行
 MOV  AL,0                 ; プログラムの実行を示す
 MOV  CX,SEG Parms         ; プログラムのパラメーター
 MOV  ES,CX
 MOV  BX,OFFSET Parms      : ES:BXはパラメーター・ブロックを指す
 MOV  CX,SEG PName         ; プログラム名
 MOV  DS,CX
 MOV  DX,OFFSET PName      ; DS:DXはプログラム名を指す
 MOV  WORD PTR StackSave+0,SP ; スタック・ポインターの保存
 MOV  WORD PTR StackSave+2,SS
 INT  21H                  ; DOSにリクエストを発行
 JC   Error                ; AX内にエラー・コード
; Note: すべてのレジスター(CS:IPを除く)は破壊される

; プログラムはここから実行

 CLI                       ; スタック使用から保護
 MOV  SS,WORD PTR StackSave+2 ; スタック・ポインターの復元
 MOV  SP,WORD PTR StackSave+0
 STI                       ; 割り込み許可
 MOV  AH,4DH               ; ファンクション・コール-リターン・
                           ; コードの取得
 INT  21H                  ; DOSにリクエストを発行
 MOV  RetCode,AX           ; リターン・コードの保存
```

```
; オーバーレイのロード
 MOV  AH,4BH                    ; ファンクション・コール-プログラムの実行
 MOV  AL,3                      ; オーバーレイのロードを示す
 MOV  CX,SEG OParms             ; オーバーレイ・パラメーター
 MOV  ES,CX
 MOV  BX,OFFSET OParms          ; ES:BXはパラメーター・ブロックを指す
 MOV  CX,SEG PName              ; オーバーレイ名
 MOV  DS,CX
 MOV  DX,OFFSET PName           ; DS:DXオーバーレイ・ファイル名を指す
 INT  21H                       ; DOSにリクエストを発行
 JC   Error                     ; AX内にエラー・コード

----

PName      DB    64 Dup (0)     ; ASCIIZ名
                                ; 例  : "c:¥dir¥File.ext",0
Parms      LABEL WORD           ; プログラムのパラメーター
Env@       DW    ?              ; 環境セグメントのアドレス
                                ;   0000Hの値はcopy EXEC'er環境を示す
Cmd@       DD    ?              ; コマンド・ライン・アドレス
FCB1@      DD    ?              ; 新PSP+5CHを設定するFCBイメージ
FCB2@      DD    ?              ; 新PSP+6CHを設定するFCBイメージ


StackSave  DD    ?              ; スタック・ポインター保存領域
RetCode    DW    ?              ; プログラム・リターン・コード
                                ; (詳細はファンクション・コード4DH参照)
OParms     LABEL WORD           ; オーバーレイ・パラメーター
Load@      DW    ?              ; オーバーレイ・ロード・セグメント・
                                ; アドレス
RelocFactor DW   ?              ; 適用する再配置要素(.EXEファイル用)
```

解説

エラー・コードはAXで返されます。エラー・クラス、推奨される対処方法、位置についての追加情報のためには、ファンクション・コール59H(拡張エラーの取得)を発行してください。ファンクション・コール59Hから返されるコードについての詳細情報は、B-6ページの『エラーに対する応答』とB-6ページの『拡張エラー・コード』を参照してください。

次の機能値がAL内で許されています。

機能値	解説
00H	プログラムをロードし実行します。プログラム・セグメント・プリフィックスがプログラムのために設定されます。終了およびCtrl-BreakアドレスはEXECシステム・コールの後、その命令に設定されます。 **注:** 制御が戻されるとすべてのレジスターはスタックも含めて変更されます。ユーザーは処理を進める前にSS、SPと他の必要なレジスターを復元しなければなりません。
03H	ロードします。プログラム・セグメント・プリフィックスは作成せず、実行を開始しません。これはプログラム・オーバーレイをロードする時に役に立ちます。

プロセスのオープンしているすべてのファイルは、ファイルが継承ビットを１に設定されてオープンされていない限り、EXECの後、新しく作成されたプロセスに複製されます。これは親プロセスが標準入力、出力、補助およびプリンター装置の意味を制御していることを示します。たとえば、親は一連のレコードをファイルに書く、標準入力としてファイルをオープンする、標準出力としてリスティング・ファイルをオープンする、それから標準入力から入力を受け取り、ソート・プログラムを実行し、標準出力に書き込むことができます。

継承(または親から複写されたもの)も**環境**です。これはさまざまなコンフィギュレーション・パラメーターを意味する文字列(合計で32KB以下)のブロックです。次に(常にパラグラフ境界上の)環境の形式を示します。

バイトASCIIZ文字列１
バイトASCIIZ文字列２
...
バイトASCIIZ文字列ｎ
0のバイト

たとえば、環境文字列は次の形式をしています。

パラメーター＝値

環境で０のバイトに続くのは、他の続く文字列数を示すWORDです。これに続くのは、子プロセスに渡されるDS:DXファイル名のコピーです。たとえば、文字列VERIFY=ONが渡せます。環境アドレスの０の値は、新しく作成されたプロセスに、変更されていないオリジナルの環境を継承します。環境のセグメント・アドレスは、コールされるプログラムのプログラム・セグメント・プリフィックスのオフセット2CHに置かれます。

エラー・コードはAXで返されます。返されるコードについての詳細情報は、B-6ページの『エラーに対する応答』とB-6ページの『拡張エラー・コード』を参照してください。

注: プログラムが制御を受け取った時、すべての利用できるメモリーは、それに割り当てられています。呼び出しているプログラムをロードできるように、ユーザーは幾らかのメモリーを解放(ファンクション・コール4AHを参照)しなければなりません。通常、ユーザーは必要なメモリーを最小にして、残りを解放します。

4CH —
プロセスの終了(EXIT)

目的

現行プロセスを終了し、呼び出したプログラムに制御を移します。

例

```
          MOV     AH,4CH            ; ファンクション・コール-プロセスの終了
          MOV     AL,ErrorCode      ; ERRORLEVELの設定
          INT     21H               ; DOSにリクエストを発行

  ----

  ErrorCode       DB      ?         ; エラー・コード(COMMAND.COMで実行したら
                                    ; ERRORLEVELを設定)
```

解説

さらに、リターン・コードが送られます。リターン・コードは、バッチのサブコマンドIFとERRORLEVELおよびWaitファンクション・コール4DHによって質問されます。
このプロセスによって、オープンされたすべてのファイルはクローズします。

4DH —
サブプロセスからのリターン・コードの取得(WAIT)

目的

ファンクション・コール4CHまたはファンクション・コール31Hで、他のプロセスによって指定されたリターン・コードを取得します。Exitコードは１度だけ返します。

例

```
        MOV     AH,4DH          ; ファンクション・コール-リターン・
                                ; コードの取得
        INT     21H             ; DOSにリクエストを発行
        MOV     RetCode,AX      ; リターン・コードの保存

  RetCode       LABEL   WORD    ; プログラム・リターン・コード
  ExitCode      DB      ?       ; ERRORLEVEL値
  ExitType      DB      ?       ; 終了に使用される方式:
                                ;   00H - 通常終了
                                ;   01H - Ctrl-Breakによる終了
                                ;   02H - 重大装置エラーによる終了
                                ;   03H - コール31Hによる終了
```

解説

終了コードの下位バイトは、終了ルーチンによって送られる情報です。

4EH —
最初に一致するファイルを見つける(FIND FIRST)

目的

指定されたファイル仕様と一致する最初のファイル名を見つけます。

例

```
        MOV     AH,1AH          ; ファンクション・コール-DTAアドレス
                                ; の設定
        MOV     CX,SEG DTA      ; 見つけるファイルのバッファーにアドレス
        MOV     DS,CX
        MOV     DX,OFFSET DTA
        INT     21H             ; DOSにリクエストを発行
        MOV     AH,4EH          ; ファンクション・コール-ASCIIZ最初に
                                ; 見つける
        MOV     CX,SEG FName    ; ディレクトリーまたはファイル名
        MOV     DS,CX
        MOV     DX,OFFSET FName ; DS:DXはASCIIファイル名を指す
        MOV     CX,Attribute    ; 一致アトリビュートの設定
        INT     21H             ; DOSにリクエストを発行
        JC      Error           ; AX内にエラー・コード

----

DTA           LABEL   BYTE              ; リターン情報を見つける
              DB      21 DUP(0)         ; 継続探索用にDOS/Vのために
                                        ; 予約済み
FileAttr      DB      ?                 ; 一致したファィルの下位バイト
FileTime      DW      ?                 ; 時間をファイル
FileDate      DW      ?                 ; 日付をファイル
FileSize      DD      ?                 ; サイズをファイル
FileNameExt   DB      "????????.???",0  ; ファイル名とエクステンション
FName         DB      64 DUP (0)        ; ASCIIZ名
                                        ;   例: "c:¥dir¥*.*",0
Attribute     DW      ?                 ; ファイル属性の選択
                                        ; 次の組み合わせ:
                                        ;  0002H=隠しファイル
                                        ;  0004H=システム
                                        ;  0008H=ボリューム・ラベル
                                        ;  0010H=ディレクトリー
```

注:

1. 属性が０の場合、通常のファイル・エントリーのみが見つけられます。ボリューム・ラベル、サブディレクトリー、隠しファイルおよびシステム・ファイルのためのエントリーは返されません。
2. 属性フィールドが隠しファイル、システム・ファイルまたはディレクトリー・エントリーに設定されている場合、それは包含探索と考えられます。すべての通常ファイル・エントリーに加えて、指定された属性と一致するすべてのエントリーが返されます。ボリューム・ラベルを除くすべてのディレクトリー・エントリーを見るためには、属性バイトを隠しファイル+システム+ディレクトリー(３ビットすべてをオン)に設定します。
3. 属性フィールドがボリューム・ラベルに設定されている場合、それは排他的探索と考えられ、ボリューム・ラベル・エントリーのみが返されます。

解説

DS:DX内のファイル名は、?や*のグローバル・ファイル名文字を含むことができます。ASCIIZ文字列はネットワーク・パスを含めません。属性ビットが探索にどのように使用されるかについては、ファンクション・コール11H(最初のエントリーの探索)を参照してください。エラー・コードはAXで返されます。エラー・クラス、推奨される対処方法、位置についての追加情報のためには、ファンクション・コール59H(拡張エラーの取得)を発行してください。ファンクション・コール59Hから返されるコードについての詳細情報は、B-6ページの『エラーに対する応答』とB-6ページの『拡張エラー・コード』を参照してください。

注: 見つけられたファイルの名前とエクステンションは、ASCIIZ文字列として返されます。すべてのブランクは、名前とエクステンションから取り除かれ、エクステンションがあれば、前にピリオドが置かれます。

4FH —
次に一致するファイルを見つける(FIND NEXT)

目的

以前のFind FirstやFind Nextファンクション・コールで指定された名前と一致する、次のディレクトリー・エントリーを見つけます。

例

```
        MOV     AH,1AH          ; ファンクション・コール-DTAアドレス設定
        MOV     CX,SEG DTA      ; 見つけるファイルのアドレス・バッファー
        MOV     DS,CX
        MOV     DX,OFFSET DTA
        INT     21H             ; DOSにリクエストを発行
        MOV     AH,4FH          ; ファンクション・コール-次を見つける
        INT     21H             ; DOSにリクエストを発行
        JC      Error           ; AX内にエラー・コード

  ----

  DTA           LABEL   BYTE          "  ; リターン情報を見つける
                DB      21 DUP(0)     "  ; DOSの継続探索用に予約済み
                                      "  ; Find Firstや以前のFind Next
                                         ; によって設定
  FileAttr      DB      ?             "  ; 見つかったファイル属性の下位バイト
  FileTime      DW      ?             "  ; 時間をファイル
  FileDate      DW      ?             "  ; 日付をファイル
  FileSize      DD      ?             "  ; サイズをファイル
  FileNameExt   DB      "????????.???",0 ; ファイル名とエクステンション
```

解説

一致するファイルが見つかると、DTAはファンクション・コール4EH(最初に一致するファイルを見つける)で述べられたように設定されます。一致するファイルが見つからなければ、エラー・コードが返されます。

エラー・コードはAXで返されます。エラー・クラス、推奨される対処方法、位置についての追加情報のためには、ファンクション・コール59H(拡張エラーの取得)を発行してください。

54H —
ベリファイ(Verify)設定状況の取得

目的

ベリファイ・フラグの値を返します。

例

```
        MOV     AH,54H          ; ファンクション・コール-ベリファイ
                                ; 設定の取得
        INT     21H             ; DOSにリクエストを発行
        MOV     VERIFY,AL       ; VERIFY状態の保存
  -----

 VERIFY         DB      ?       ; VERIFY状態：
                                ; 0 = OFF
                                ; 1 = ON
```

解説

リターンの際に、ALはベリファイがOFFならば00Hを返し、ONならば01Hを返します。ベリファイ・スイッチはファンクション・コール2EH(ベリファイ・スイッチの設定と解除)で設定できます。

56H —
ファイル名の変更

目的

指定されたファイルの名前を変更します。

例

```
        MOV     AH,56H              ; ファンクション・コール-ASCIIZ
                                    ; ファイル名変更
        MOV     CX,SEG FName        ; ファイル名
        MOV     DS,CX
        MOV     DX,OFFSET FName     ; DS:DXはオリジナル名を指す
        MOV     CX,SEG NewName      ; 新ファイル名
        MOV     ES,CX
        MOV     DI,OFFSET NewName   ; ES:DIは名前変更を指す
        INT     21H                 ; DOSにリクエストを発行
        JC      Error               ; AX内にエラー・コード

  ----

  FName         DB      64 DUP (0)    ; ASCIIZ 名
                                      ;    例: "c:¥dir¥abc.1st",0
  NewName       DB      64 DUP (0)    ; ASCIIZ 名
                                      ;    例: "¥dir¥xyz.1st",0
```

解説

ドライブがNewName文字列に含まれている場合、それは指定されたドライブ、またはName文字列に含まれたドライブと同じでなければなりません。ディレクトリー・パスは同じである必要はなく、ファイルを他のディレクトリーに移動したり、プロセスの中で名前を変更することができます。ディレクトリー名は変更できますが、移動はできません。?や*のグローバル・ファイル名文字はファイル名に使用できません。

エラー・コードはAXで返されます。エラー・クラス、推奨される対処方法、位置についての追加情報のためには、ファンクション・コール59H(拡張エラーの取得)を発行してください。ファンクション・コール59Hから返されるコードについての詳細情報については、B-6ページの『エラーに対する応答』とB-6ページの『拡張エラー・コード』を参照してください。

ネットワーク・アクセス権: 作成アクセス権が必要。

57H —
ファイルの日付および時刻の取得と設定

目的

ファイルの日付および時刻の取得と設定をします。

例

```
 ; ファイルの日付および時刻を取得

      MOV     AH,57H          ; ファンクション・コール-日付と時刻の取得
      MOV     AL,0            ; 取得を示す
      MOV     BX,Handle       ; ファイルの選択
      INT     21H             ; DOSにリクエストを発行
      JC      Error           ; AX内にエラー・コード
      MOV     FileTime,CX     ; 時刻の保存
      MOV     FileDate,DX     ; 日付の保存

; ファイルの日付および時刻の設定

      MOV     AH,57H          ; ファンクション・コール-日付と時刻の設定
      MOV     AL,1            ; 設定を示す
      MOV     BX,Handle       ; ファイルの選択
      MOV     CX,FileTime     ; 時刻の設定
      MOV     DX,FileDate     ; 日付の設定
      INT     21H             ; DOSにリクエストを発行
      JC      Error           ; AX内にエラー・コード

----

Handle        DW      ?       ; ファイル・ハンドル(OpenまたはCreateから)
FileTime      DW      ?       ; 時間をファイル
FileDate      DW      ?       ; 日付をファイル
```

解説

日付および時刻形式は、3-5ページの『ファイルの作成時刻と最新の変更時刻』に述べられているディレクトリー・エントリーと同じです。

エラー・コードはAXで返されます。エラー・クラス、推奨される対処方法、位置についての追加情報のためには、ファンクション・コール59H(拡張エラーの取得)を発行してください。ファンクション・コール59Hから返されるコードについての詳細情報については、B-6ページの『エラーに対する応答』とB-6ページの『拡張エラー・コード』を参照してください。

ネットワーク・アクセス権:　作成アクセス権が必要。

59H —
拡張エラーの取得

目的

エラー・クラス、場所、推奨される対処方法のような追加のエラー情報を返します。

例

```
        PUSH    DX               ; レジスターの保存
        PUSH    SI
        PUSH    DI
        PUSH    ES
        PUSH    DS
        MOV     AH,59H           ; ファンクション・コール-拡張エラーの取得
        MOV     BX,0             ; バージョン0情報
        INT     21H              ; DOSにリクエストを発行
        POP     DS               ; レジスターの復元
        POP     ES
        POP     DI
        POP     SI
        POP     DX
        MOV     ExtError,AX      ; エラー・コードの保存
        MOV     ErrorClass,BH    ; エラー・クラスの保存
        MOV     ErrorAction,BL   ; 対処方法の保存
        MOV     ErrorLocation,CH; エラー場所の保存

  ----

  ExtError      DW      ?        ; DOS拡張エラー
  ErrorClass    DB      ?        ; エラー・クラス
  ErrorAction   DB      ?        ; 推奨される対処方法
  ErrorLocation DB      ?        ; 起こったシステム領域
```

解説

このファンクション・コールは、リターン・コードに加えてエラー・クラス、位置、推奨される対処方法を返します。次に示すものからこのファンクション・コールを使用してください。

- 割り込み24Hエラー・ハンドラー
- キャリー・ビットにエラーを返す割り込み21Hファンクション・コールFFHを返すFCBファンクション・コール
- リターンに際しては、DX、SI、DI、ES、CLおよびDSレジスターの内容は破壊されます。

キャリー・ビットで返るエラー

キャリー・フラグを設定することで、エラーを示すファンクション・コールにおいて、ファンクション・コール59Hを実行する正しい方法を次に示します。

- レジスターをロードします。
- 割り込み21Hを発行します。
- キャリーが設定されなければ操作を続けます。
- エラー・コードを無視し、追加情報を得るためにファンクション・コール59Hを発行します。
- 行うべき推奨される対処方法を決定するためにＢＬ内の値を使用します。

AL内のエラー状況

ALにFFHを設定することで、エラーを示すファンクション・コールにおいて、ファンクション・コール59Hを実行する正しい方法を次に示します。

- レジスターをロードします。
- 割り込み21Hを発行します。
- エラーがALで報告されなければ操作を続けます。
- エラー・コードを無視し、追加情報を得るためにファンクション・コール59Hを発行します。
- 行うべき推奨される対処方法を決定するためにBLの値を使用します。

5AH —
ユニーク・ファイル作成

目的

ユニーク・ファイル名を生成し、指定されたディレクトリーにファイルを作成します。

例

```
          MOV     AH,5AH                 ; ファンクション・コール-ユニーク・
                                         ; ファイルの作成
          MOV     CX,SEG DirName         ; デイレクトリー名
          MOV     DS,CX
          MOV     DX,OFFSET DirName
          MOV     CX,Attribute           ; ファイル属性
          INT     21H                    ; DOSにリクエストを発行
          JC      Error                  ; AX内にエラー・コード
          MOV     Handle,AX              ; 次の操作のためファイル・
                                         ; ハンドルを保存

    ----

    DirName       DB      "?? .. ??\",0  ; ASCIIZ名
                  DB      "????????.???"
                                         ;   入力例: "c:¥dir¥",0
                                         ;   出力例: "c:¥dir¥file",0
    Handle        DW      ?              ; ファイル・ハンドル
    Attribute     DW      ?              ; ファイル属性の選択
```

解説

エントリーで、AHは5AHです。ファイルは読み取り／書き込みアクセスを持つ互換モードでオープンされます。読み取り／書き込みポインターはファイルの最初のバイトに設定され、AXはファイル・ハンドルです。そしてファイル名は、DS:DXに指定されたパスに追加されます。

このファンクション・コールはユニークな名前を生成し、指定されたディレクトリー内に新しいファイルを作成しようとします。そのファイルがすでにディレクトリーに存在する場合、他のユニークな名前が生成され、プロセスが繰り返されます。テンポラリー・ファイルを必要とするプログラムは、ユニーク・ファイル名を生成するために、このファンクション・コールを使用してください。

エラー・コードはAXで返されます。エラー・クラス、推奨される対処方法、位置についての追加情報のためには、ファンクション・コール59H(拡張エラーの取得)を発行してください。ファンクション・コール59Hから返されるコードについての詳細情報につ

いては、B-6ページの『エラーに対する応答』とB-6ページの『拡張エラー・コード』を参照してください。

注: このファンクション・コールを使用して作成されたファイルは、プログラム終了時には自動的に削除されません。

ネットワーク・アクセス権: 作成アクセス権が必要。

5BH —
新しいファイルの作成

目的

新しいファイルを作成します。

例

```
        MOV     AH,5BH          ; ファンクション・コール-新しいファイル
                                ; の作成
        MOV     CX,SEG FName    ; ファイル名
        MOV     DS,CX
        MOV     DX,OFFSET FName
        MOV     CX,Attribute
        INT     21H             ; DOSにリクエストを発行
        JC      Error           ; AX内にエラー・コード
        MOV     Handle,AX       ; 次の操作のためファイル・ハンドルを保存

  ----

  FName         DB      64 DUP (0)    ; ASCIIZ名
                                      ;   例: "c:¥dir¥file",0
  Handle        DW      ?             ; ファイル・ハンドル
  Attribute     DW      ?             ; ファイル属性の選択
```

解説

このファンクション・コールは、ファイル名がすでに存在する場合には失敗(Fail)することを除いて、ファンクション・コール3CH(Create)と同じです。ファイルは、読取りと書込みに対して互換モードで作成され、読み取り／書き込みポインターはファイルの最初のバイトに設定されます。

エラーコードはAXで返されます。エラー・クラス、推奨される対処方法、位置についての追加情報のためには、ファンクション・コール59H(拡張エラー取得)を発行してください。ファンクション・コール59Hから返されるコードについての詳細情報については、B-6ページの『エラーに対する応答』とB-6ページの『拡張エラー・コード』を参照してください。

ネットワーク・アクセス権:　作成アクセス権が必要。

5CH —
ファイル・アクセスのロックまたはロック解除

目的

オープンされたファイルの一定範囲のバイトを、ロックまたはその解除をします。このファンクション・コールは、ネットワーク環境でデータベースの完全性を維持するために有効な、データ・ベース・サービスを提供します。

例

```
; 一定範囲のロック

      MOV     AH,5CH                    ; ファンクション・コール-ロック／
                                        ; ロック解除ファイルのアクセス
      MOV     AL,0                      ; ロックを示す
      MOV     BX,Handle                 ; ファイル選択
      MOV     DX,WORD PTR Position+0    ; 位置設定
      MOV     CX,WORD PTR Position+2
      MOV     DI,WORD PTR Llength+0     ; 長さ設定
      MOV     SI,WORD PTR Llength+2
      INT     21H                       ; DOSにリクエストを発行
      JC      Error                     ; AX内にエラー・コード

; 一定範囲のロック解除

      MOV     AH,5CH                    ; ファンクション・コール-
                                        ; ファイル・アクセスのロック／
                                        ; ロック解除
      MOV     AL,1                      ; ロック解除を示す
      MOV     BX,Handle                 ; ファイル選択
      MOV     DX,WORD PTR Position+0    ; 位置の設定
      MOV     CX,WORD PTR Position+2
      MOV     DI,WORD PTR Llength+0     ; 長さの設定
      MOV     SI,WORD PTR Llength+2
      INT     21H                       ; DOSにリクエストを発行
      JC      Error                     ; AX内にエラー・コード

----

Handle        DW      ?                 ; ファイル・ハンドル
Position      DD      ?                 ; 範囲の開始
Llength       DD      ?                 ; 範囲の長さ
```

解説

ロック／ロック解除のファンクション・コールは、ファイルが読み取り拒否または非拒否共用モードを使用してオープンされている時にのみ使用すべきです。これらのモードはネットワーク・ディスク上のファイルにアクセスする時、データのローカル・バッファリングを行いません。

AL=00H　ロック

ファイル領域への、他のプロセスから読み取り／書き込みアクセスを排除する、簡単なメカニズムを提供します。他のプロセスがそのような領域に、読み取りまたは書込みをしようとすると、そのシステム・コールは、IOCtlによって設定された、システム再試行カウントで指定される回数だけ再試行されます。これらの再試行の後成功しなければ、一般障害エラーが生成され、その状態を知らせます。再試行の間の時間の長さだけでなく、再試行回数もファンクション・コール440BH(共用再試行回数の変更)を使用して変更できます。

推奨される対処方法は、エラー・クラス、位置、推奨される対処方法の追加エラー・コードを得るために、ファンクション・コール59H(拡張エラーの取得)を発行することです。ロックされる領域は論理ファイルのどこでも可能です。ファイルの終わり(EOF)を越えてロックすることは、エラーではありません。その場合短い時間だけ領域がロックされると考えられます。ハンドルの複製は、ロックされた領域へのアクセスを複製します。ロックされた領域へのアクセスは、EXECシステム・コールを通じては複製されません。ファイルをオープンのまま終了し、そのファイルにロックを発行していた場合は、結果が予想できなくなります。

INT23HまたはINT24Hを使用して取り消されるプログラムは、これらの割り込みをトラップし、終了前にロックを解放しなければなりません。ロックを使用する適当な方法としては、読み書き拒否アクセスに頼らず、希望する領域のロックを試みて、エラー・コードを検査することです。

AL=01H　ロック解除

ロック解除は、ロック・システム・コールで発行されたロックを解除します。指定された領域は、以前のロックで指定された領域と正確に同じでなければなりません。まだ有効なロックでファイルをクローズすると、結果が予想できなくなります。ファイルをオープンのまま終了し、そのファイルにロックを発行していた場合も、結果が予想できなくなります。

INT23HまたはINT24Hを使用して打ち切られるプログラムは、これらの割り込みをトラップし、終了前にロックを解放しなければなりません。ロックを使用する適当な方法としては、読み書き拒否アクセスに頼らず、希望する領域のロックを試みて、エラー・コードを検査することです。

エラー・コードはAXで返されます。エラー・クラス、推奨される対処方法、位置についての追加情報のためには、ファンクション・コール59H(拡張エラーの取得)を発行してください。ファンクション・コール59Hから返されるコードについては、B-6ペー

ジの『エラーに対する応答』とB-6ページの『拡張エラー・コード』を参照してください。

5E00H —
機械名の取得

目的

ローカル・コンピューターの文字IDを返します。

例

```
MOV    AX,SEG CNAME      ; 名前バッファー
MOV    DS,AX
MOV    DX OFFSET CNAME
MOV    AX,5E00H          ; ファンクション・コール-
                         ; 機械名の取得
INT    21H               ; DOSにリクエストを発行
JC     Error             ; AX内にエラー・コード
MOV    NameFlag,CH       ; 名前番号インジケーターの保存
MOV    NameID,CL         ; NETBIOS名前番号の保存

  ----

CName          DB     "???????????????",0    ; ASCIIZ コンピューター名
NameFlag       DB     ?                      ; 0 = 名前は設定されていない
                                             ; 1 = 名前は設定されている
NameID         DB     ?                      ; NETBIOS 名前番号
```

解説

「機械名の取得」は、コール側に現行コンピューター名を返します。コンピューター名は、スペースで埋められた15文字バイト文字列で、00Hバイトが続きます。コンピューター名が設定されていないと、CHレジスターに00Hが返され、CLレジスターの値は無効となります。このファンクション・コールが正しく実行されるためにはネットワーク・プログラムがロードされていなければなりません。

5E02H —
プリンター・セットアップの設定

目的

プリンター・ファイルの初期文字列を指定します。

例

```
        MOV     AX,5E02H        ; ファンクション・コール-
                                ; プリンター・セットアップの設定
        MOV     BX,Index        ; リダイレクション・リスト・インデックス
        MOV     CX,size         ; 文字列サイズ
        MOV     SI,SEG String   ; 文字列バッファー
        MOV     DS,SI
        MOV     SI,OFFSET String
        INT     21H             ; DOSにリクエストを発行
        JC      Error           ; AX内にエラー・コード

  ----

  Index         DW     ?        ; リダイレクション・リスト・インデックス
  Size          DW     N        ; 文字列サイズ(最大 64)
  String        DB     N DUP(?) ; プリンター・セットアップ文字列
```

解説

特定のネットワーク・プリンターへ送られるすべてのファイルの前に、指定された文字列が付けられます。「プリンター・セットアップの設定」では、１台のプリンターに複数ユーザーが、自分自身の操作モードを指定できます。BXはファンクション・コール5F20H(リダイレクション・リスト・エントリーの取得)で使用されるものと同じインデックスに設定されます。印刷リダイレクションが止まっている、またはネットワーク・プログラムがロードされていないと、エラー・コードが返されます。

エラー・コードはAXで返されます。エラー・クラス、推奨される対処方法、位置についての追加情報のためには、ファンクション・コール59H(拡張エラーの取得)を発行してください。ファンクション・コール59Hから返されるコードの詳細情報については、B-6ページの『エラーに対する応答』とB-6ページの『拡張エラー・コード』参照してください。

重要:リダイレクション・リストが走査される時間と、ファンクション・コール5E02H(プリンター・セットアップの設定)が発行される時間の間に、ファンクション・コール5F03H(装置のリダイレクション)またはファンクション・コール5F04H(リダイレクションの取り消し)が発行されると、リダイレクション・インデックス値が変更されるかもしれません。そのため「リダイレクション・リストの取得」を発行した後すぐに、「プリンター・セットアップの設定」を発行するようにしてください。

5E03H —
プリンター・セットアップの取得

目的

プリンター・ファイルのプリンター・セットアップ文字列を返します。

例

```
        MOV     AX,5E03H        ; ファンクション・コール-
                                ; プリンター・設定の取得
        MOV     BX,Index        ; リダイレクション・リスト・インデックス
        MOV     CX,SEG String   ; 文字列バッファー
        MOV     ES,CX
        MOV     DI,OFFSET String
        INT     21H             ; DOSにリクエストを発行
        JC      Error           ; AX内にエラー・コード
        MOV     Ssize, CX       ; 文字列サイズの保存

  ----

  Index         DW    ?         ; リダイレクション・リスト・インデックス
  Ssize         DW    ?         ; 文字列サイズ
  String        DB    64 DUP(?) ; プリンター・セットアップ文字列
```

解説

このファンクション・コールは、ファンクション・コール5E02H(プリンター・セットアップの設定)を使用して指定されたプリンター・セットアップ文字列を返します。セットアップ文字列は、特定プリンターに送られるすべてのファイルに付けられます。BXの値は、ファンクション・コール5F02H(リダイレクション・リストの取得)で発行されたものと同じインデックスに設定されます。エラー・コード 1 (無効な機能番号)は、ネットワーク・プログラムがロードされていないと返されます。

エラー・コードはAXで返されます。エラー・クラス、推奨される対処方法、位置についての追加情報のためには、ファンクション・コール59H(拡張エラーの取得)を発行してください。ファンクション・コール59Hから返されるコードについての詳細情報については、B-6ページの『エラーに対する応答』とB-6ページの『拡張エラー・コード』を参照してください。

重要: リダイレクション・リストが走査される時間と、ファンクション・コール5E03H(プリンター・セットアップの取得)が発行される時間の間に、ファンクション・コール5F03H(装置のリダイレクション)またはファンクション・コール5F04H(リダイレクションの取り消し)が発行されると、リダイレクション・インデックス値が変更されるかもしれません。そのため、「リダイレクション・リストの取得」を発行した後すぐに「プリンター・セットアップの取得」を発行するようにしてください。

5F02H —
リダイレクション・リスト・エントリーの取得

目的

非ローカル・ネットワークの割り当てを返します。

例

```
     MOV      BX,0              ; リストの最初で開始
Get_Loop:                       ; 次のエントリーの取得
     MOV      Index,BX          ; リダイレクション・リスト・インデックス
     MOV      AX,5F02H          ; ファンクション・コール-
                                ; リダイレクション・リスト・エントリー
                                ; の取得
     MOV      SI,SEG device     ; "PRN" は可能
     MOV      DS,SI
     MOV      SI,OFFSET device  ; DOS:SIはローカル名を指す
     MOV      DI,SEG info
     MOV      ES,DI
     MOV      DI,OFFSET inf     ; ES:DIネットワーク名のバッファー・
                                ; アドレスを指す
     PUSH     BX
     PUSH     DX                ; レジスター保存
     PUSH     BP
     INT      21H               ; DOSにリクエストを発行
     POP      BP                ; レジスターの復元
     POP      DX
     JC       CheckEnd          ; AX内にエラー・コード
     MOV      Status,BH         ; 状況の保存
     MOV      Type,BL           ; タイプの保存
     MOV      UserParm,CX       ; ユーザー・パラメーターの保存
     POP      BX
     INC      BX                ; 次のエントリーの設定
     JMP      Get_Loop

CheckEnd:
      POP     BX                ; 状態をバランス
      CMP     AX,18             ; リストの終わり?
      JNE     Error             ; いいえ!
```

```
----

Index       DW     ?                ; リダイレクション・リスト・
                                    ; インデックス(0基準)
Device      DB     128 DUP(?)       ; ASCIIZ装置名
                                    ;  例: "LPT1",0 "LPT1",0
                                    ;       "A:",0
Status      DB     ?                ; 装置状況
                                    ; Bit 0=0: 装置はOK
                                    ; Bit 0=1: 装置エラー
                                    ; Bit 7-1 予約済み
UserParm    DW     ?                ; ユーザー・パラメーター
Type        DB     ?                ; 装置のタイプ
                                    ; 3 = NET USE装置ice
                                    ; 4 = NET USEドライブ
Info        DB     128 DUP(?)       ; NET USEネットワーク・パス
                                    ;   例: "¥¥MYNODE¥CDRIVE",0CDRIVE",0
```

解説

「リダイレクション・リスト・エントリーの取得」ファンクション・コールは、ファンクション・コール5F03H(装置のリダイレクション)で作成されたネットワーク・リダイレクションのリストを返します。各コールは、１つのリダイレクションを返します。それによりBXは、リストに入るたびに一度に１ずつ増加します。リストの内容は、コールの間に変更されます。リストの終わりは、エラー・コード18(これ以上ファイルがない)によって検出されます。エラー・コード１(無効な機能番号)は、ネットワークがロードされないと返されます。

ディスクまたはプリント・リダイレクションのどちらかが止まっていると、この機能は無効です。

エラー・コードはAXで返されます。エラー・クラス、推奨される対処方法、位置についての追加情報のためには、ファンクション・コール59H(拡張エラーの取得)を発行してください。ファンクション・コール59Hから返されるコードについての詳細情報については、B-6ページの『エラーに対する応答』とB-6ページの『拡張エラー・コード』を参照してください。

5F03H —
装置のリダイレクト

目的

リダイレクター／サーバー接続を行います。

例

```
          MOV     AX,5F03H              ; ファンクション・コール-
                                        ; 装置のリダイレクト
          MOV     SI,SEG Device         ; 装置バッファー
          MOV     DS,SI
          MOV     SI,OFFSET Device
          MOV     DI,SEG Net_Path       ; 情報バッファー
          MOV     ES,DI
          MOV     DI,OFFSET Net_Path
          MOV     BL,Type               ; タイプの設定
          MOV     CX,UserParm           ; ユーザー・パラメーターの設定
          INT     21H                   ; DOSにリクエストを発行
          JC      Error                 ; AX内にエラー・コード
    ----

    Device        DB      "....",0      ; ASCIIZ装置名
                                        ;   例: "LPT1",0
                                        ;        "A:",0
    UserParm      DW      ?             ; ユーザー・パラメーター
    Type          DB      ?             ; 装置タイプ
                                        ; 3 = NET USE装置
                                        ; 4 = NET USEドライブ
    Net Path      DB     128 DUP(0)
```

解説

このコールは、ネットワークのために現行ディレクトリーを定義し、ネットワーク・プリンターのリダイレクションを定義します。

- BL=3の場合、ソース(割り当て元)はプリンターを指定し、宛先(割り当て先)はネットワーク・パスを指定します。そしてCXレジスターには、DOS/Vがプログラマーのために維持しているワードがあります。ネットワーク・プログラムとの互換性のために、CXは０に設定すべきです。０以外の値はネットワーク・プログラムのために予約されています。このワードは、ファンクション・コール5F02H(リダイレクション・リストの取得)から検索されます。指定されたプリンターに送られるすべての出力は、バッファーに入れられ、その装置のリモート・プリンター・スプールに送られます。プリンターは、INT17Hレベルでリダイレクトされます。

ソース文字列は、それぞれ00Hで終わるPRN、LPT1、LPT2またはLPT3でなければなりません。宛先文字列は、次の形式のネットワーク名文字列を指さなければなりません。

[¥¥*computername*¥{*shortname*∨*printdevice*}]

宛先文字列は00Hで終わらなければなりません。

リモート装置へのアクセスのためのASCIIZパスワード(０から８文字)は、ネットワーク文字列の後にすぐ続けなければなりません。パスワードは00Hを最後に付けなければなりません。ヌル(長さ０)パスワードはパスワード無しと考えられます。

- BL=4の場合、ソースはドライブ文字と00Hで終わるコロンを指定し、宛先は00Hで終わるネットワーク・パスを指定します。そしてCXレジスターには、DOS/Vがプログラマーのために維持しているワードがあります。ネットワーク・プログラムとの互換性のために、CXは０に設定されなければなりません。０以外の値は、ネットワーク・プログラムのために予約されています。このワードは、ファンクション・コール5F02H(リダイレクション・リストの取得)から検索されます。ソースがドライブ文字だった場合、ドライブ文字とネットワーク・パスが関連づけられます。その後のドライブ文字へのすべての参照は、ネットワーク・パスへの参照に翻訳されます。ソースが空の文字列の場合、システムは装置をリダイレクトせずに、指定されたパスワードで宛先へのアクセスを許可しようとします。

  リモート装置へのアクセスのためのASCIIZパスワード(０から８文字)は、ネットワーク文字列の後にすぐ続けなければなりません。ヌル(長さ０)パスワードはパスワード無しと考えられます。

  エラーコードはAXで返されます。エラー・クラス、推奨される対処方法、位置についての追加情報にはファンクション・コール59Hを発行してください。ファンクション・コール59H(拡張エラーの取得)から返されるコードについての詳細情報は、B-6ページの『エラーに対する応答』とB-6ページの『拡張エラー・コード』を参照してください。

注:

1. このファンクション・コールからリダイレクトされた装置は、NET USEコマンドによって表示されません。
2. ディスク・リダイレクションが止まっている間にドライブをリダイレクトしようとした場合、またはプリンター・リダイレクションが止まっている間にプリンターをリダイレクトしようとした場合、エラーが返されます。

5F04H —
リダイレクションの取り消し

目的

前のリダイレクションを取り消します。

例

```
        MOV     AX,5F04H        ; ファンクション・コール-
                                ; リダイレクションの取り消し
        MOV     SI,SEG Device   ; 装置バッファー
        MOV     DS,SI
        MOV     SI,OFFSET Device
        INT     21H             ; DOSにリクエストを発行
        JC      Error           ; AX内にエラー・コード

  ----

  Device        DB      "....",0        ; ASCIIZ装置名
                                        ;  例: "LPT1",0
                                        ;      "A:",0
                                        ;      "¥¥Computer¥Path",0
```

解説

「装置のリダイレクト」ファンクション・コール(5F03H)によって作成されたリダイレクションは、「リダイレクションの取り消し」のコールで取り除かれます。バッファーがドライブ文字を示し、ドライブがネットワーク名と関連している場合、関連づけは終了し、ドライブはその物理的意味に復元されます。バッファーがＰＲＮ,ＬＰＴ１,ＬＰＴ２またはＬＰＴ３を指し、装置がネットワーク装置と関連している場合、その関連づけは終了し、装置はその物理的意味に復元されます。バッファーが00Hで終わるネットワークパスと00Hで終わるパスワードを指す場合、ローカル機械とネットワーク・ディレクトリーの間の関連づけは終了します。

ディスク・リダイレクションが止っている間に、リダイレクトしたファイル・デバイスを取り消そうとしたり、プリンター・リダイレクションが止まっている間に、リダイレクトしたプリンターを取り消そうとするとエラーが返されます。ネットワーク・プログラムがロードされていない場合、エラー・コード１(無効な機能番号)が返されます。

エラー・コードはAXで返されます。エラー・クラス、推奨される対処方法、位置についての追加情報のためには、ファンクション・コール59Hを発行してください。ファンクション・コール59H(拡張エラーの取得)から返されるコードについての詳細情報は、B-6ページの『エラーに対する応答』とB-6ページの『拡張エラー・コード』を参照してください。

62H —
プログラム・セグメント・プリフィックス(PSP)アドレスの取得

目的

プログラム・プリフィックス・アドレスを返します。

例

```
        MOV     AH,62H          ; ファンクション・コール-
                                ; プログラム・セグメント・プリフィックス・
                                ; アドレスの取得
        INT     21H             ; DOSにリクエストを発行
        JC      Error           ; AX内にエラー・コード
        MOV     PSPSeg,BX       ; PSPアドレスの保存

  ----

  PSPSeg        DW      ?       ; 自身のPSPセグメント・アドレス
```

解説

現在実行しているプロセスの内部PSPアドレスがBXで返されます。

6300H —
DBCSベクター情報の取得

目的

DBCSベクターのテーブルのアドレスを返します。

例

```
        MOV     AX, 6300H       ; DBCSベクターのテーブルのアドレスを得る
        INT     21H             ; DOSにリクエストを発行
        JC      Error           ; AX内にエラー・コード

        MOV     WORD PTR DBCS_VEC_PTR+0, SI
        MOV     WORD PTR DBCS_VEC_PTR+2、DS

  ----
```

DBCSベクター・テーブル

```
  1     DB      Start1、End1    ; DBCSベクター1
  2     DB      Start2、End2    ; DBCSベクター2
  :
  N     DB      StartN, EndN    ; DBCSベクターN
        DB         0 , 0        ; 終了マーカー
```

解説

DBCSベクター・テーブルのアドレスがDS : SI（セグメント : オフセット）に返されます。SBCS（1バイト文字セット）モードのとき、
Start 1 = End1=... = Start N = End N = 0 です。DBCS（2バイト文字セット）モードのとき、Start 1～End N は2バイト文字セットの開始バイトの範囲を示します。

65H —
拡張国別情報の取得

目的

拡張国別情報を返します。

注: 英語モードのみ有効です。

例

```
; 情報の取得

      MOV     AH,65H          ; ファンクション・コール-
                              ; 拡張国別情報の取得
      MOV     AL,InfoID       ; リクエストしたデータ／機能
                              ; (1, 2, 4, 6 or 7)
      MOV     BX,CodePage     ; 要求されたコード・ページの設定
                              ; (-1=現行、ファンクション・
                              ; コール6602Hで設定)
      MOV     CX,SizeBuffer   ; リターンする最大データ
                              ; (5以上でなければならない)
      MOV     DX,CountryID    ; 要求された国別IDの設定
                              ; (-1=現行、ファンクション・
                              ; コール38Hで設定)
      MOV     DI,SEG Buffer   ; 情報リターン・バッファー
      MOV     DS,DI
      MOV     DI,OFFSET Buffer
      INT     21H             ; DOSにリクエストを発行
      JC      Error           ; AX内にエラー・コード
-----

Buffer        LABEL   BYTE
; 形式はALの値による
; AL=1: 拡張国別情報

SizeBuffer    DW      ?       ; 返るデータ・ブロックのサイズ
InfoID        DB      ?       ; 取得する情報のタイプ
CIinfoSize    DW      ?       ; 続くデータの合計
                              ; (CX入力によって限定される)
CountryID     DW      ?       ; 選択した国別ID
CodePage      DW      ?       ; 選択したコード・ページ
; このバッファーの残りの形式はファンクション・コール38H参照
```

```
-----

; AL=2: 大文字テーブル
              DB      2       ; 「大文字テーブル」を示す
UpperCase@    DD      ?       ; 「大文字テーブル」のアドレス

; AL=4: 「ファイル大文字テーブル」
              DB      4       ; 「ファイル大文字テーブル」を示す
UpperCase@    DD      ?       ; 「ファイル大文字テーブル」のアドレス

; AL=6: 対照テーブル
              DB      6       ; 対照テーブルを示す
Collate@      DD      ?       ; 対照テーブルのアドレス

; AL=7: DBCSベクトル・テーブル
              DB      7       ; DBCSベクター・テーブルを示す
DBCS@         DD      ?       ; DBCSベクター・テーブルのアドレス
```

解説

エントリーの際、DXは拡張情報を必要とする国のIDです。ALは国のID値です。

- 国別コードとコード・ページが一致しない場合、および片方または両方が無効の場合、エラー・コード２(ファイルが見つからない)がAXで返されます。
- CXでリクエストされるサイズは５以上でなければなりません。５より小さい場合は、AXでエラー・コード１が返されます。
- 返される情報の量がCXでリクエストされたサイズより大きい場合、それは終了し、AXでは何のエラーも返されません。

注: 国別情報に関するさらに詳細な情報については、ファンクション・コール38H(国別情報の取得または設定)を参照してください。

「現行国」以外の国別情報を取得するために、NLSFUNC DOSエクステンションは導入されていなければなりません。

大文字テーブルとファイル名大文字テーブルは、130バイトの長さで、長さフィールド(２バイト)と、それに続く128個の大文字アスキー文字のための128の大文字値から構成されます。それらは次のようなレイアウトです。

```
Tsize         DW      128             ; テーブル・サイズ
Table         DB      128 DUP(?)      ; 80HからFFHの大文字バージョン
```

次に示す公式は、大文字テーブルまたはファイル名大文字テーブルの小文字(ASCII_in)に対応する、大文字のアドレスを決定するために使用することができます。

例

```
ASCII_in -(256-table_len)+table_start= address of ASCII_out
```

ここで

```
ASCII_in    = 生成される文字

table_len   = 大文字値リストの長さ(２バイト)

table_start = 大文字テーブルの開始アドレス(４バイト)

ASCII_out   = ASCII_inの大文字値
```

ASCII_inの値が、(256-table _len)以上の場合、テーブル内にASCII _に対応する大文字が存在します。(256-table_len)より小さい場合、テーブルに対応する大文字は存在しません。

対照テーブルは258バイトの長さで、長さフィールド(２バイト)と、それに適当な順で続く256個のアスキー値で構成されます。それは次のようなレイアウトです。

```
Tsize       DW     256             ; テーブル・サイズ
Table       DB     256 DUP(?)      ; 00HからFFHのソートの重み
```

DBCSベクトルの長さは可変で、長さフィールドとそれに続く１つ以上のバイト・ペアで構成されます。それは次のようなレイアウトです。

```
Tsize   DW    Nx2          ;リスト・サイズ
1       DB    Start,end    ;DBCSベクトル１
2       DB    Start,end    ;DBCSベクトル２
:
N       DB    Start,end    ;DBCSベクトルｎ
        DB    0,0          ;終了マーカー
```

66H —
グローバル・コード・ページの取得と設定

目的

この機能は、現行国のコード・ページの取得または設定を行ないます。

注: 英語モードのみ有効です。

```
MOV     AX,6601H        ; ファンクション・コール-
                        ; グローバル・コード・ページの取得
INT     21H             ; DOSにリクエストを発行
JC      Error           ; AX内にエラー・コード
MOV     GlobalCP,BX     ; グローバル・コード・ページの保存
MOV     SystemCP,DX     ; DOS システム・コード・ページの保存
```

または

```
MOV     AX,6602H        ; ファンクション・コール-
                        ; グローバル・コード・ページの設定
MOV     BX,GlobalCP     ; 新グローバル・コード・ページ
INT     21H             ; DOSにリクエストを発行
JC      Error           ; AX内にエラー・コード

----

GlobalCP    DW    ?     ; DOS国別情報の現行コード・ページ
SystemCP    DW    ?     ; DOSメッセージのコード・・ページ
                        ; しばしば現行国の省略時コード・ページ
```

解説

DOS/Vは、COUNTRY.SYSファイルから常駐の国別バッファー領域に新しいコード・ページ・データを移動します。DOS/Vコード・ページ切り替えのために準備されている(つまりCONFIG.SYSで指定されたコード・ページ切り替えデバイス・ドライバーを持っている)すべての取り付けられた装置に対して、Selectを実行するために、新しいコード・ページを使用します。どの装置も選択に失敗すると、エラー・コード65がAXで返されます。コード・ページは、現行国によって識別可能でなければなりません。そしてDOS/Vは、国別情報ファイルをオープンしてそれを読むことができなければなりません。でなければ、キャリー・フラグがリターンの際に設定され、AXは02(ファイルが見つからない)になります。

注: このファンクション・コールを使用するためには、NLSFUNCを導入しなければなりません。そしてすべての装置は、Select機能が成功するために準備されていなければなりません。

67H —
ハンドル数の設定

目的

プロセスごとに20以上のファイルを開くことを可能にします。

例

```
EntryPoint:
        MOV     AH,62H          ; ファンクション・コール-
                                ; PSPアドレスの取得
        INT     21H             ; DOSにリクエストを発行
        JC      Error           ; AX内にエラー・コード
        MOV     ES,BX           ; セグメントの設定
        MOV     AH,4AH          ; ファンクション・コール-
                                ; メモリー・ブロック・サイズの設定
        MOV     BX,paragraphs   ; サイズの設定
        INT     21H             ; DOSにリクエストを発行
        JC      Error           ; AX内にエラー・コード
        MOV     AH,67H          ; ファンクション・コール-ハンドル数の設定
        MOV     BX,NewHandles   ; 新ハンドル数の設定
        INT     21H             ; DOSにリクエストを発行
        JC      Error           ; AX内にエラー・コード

  ----

NewHandles      DW     ?                            ; このDOS/Vプロセス
                                                    ; による必要なハンドル数
ListSize        EQU    (Endofprogram-EntryPoint)    ; バイト数
Paragraphs      EQU    (ListSize / 10H)             ; パラグラフ数
End_of_program  LABEL  BYTE                         ; これに最後に使用
                                                    ; した位置
```

解説

この割り込みに可能なファイル・ハンドルの最大数は64KBです。許可されるハンドルの指定された数が、許可されている現在のハンドル数より小さい場合、指定された数を超えたすべてのハンドルが閉じられた後にのみ、指定の数が現行となります。指定数が20より小さい場合、その数は20とみなされます。データ・ベース・アプリケーションは、ハンドルを交換する必要を減らすために、この機能を使用することができます。

拡張されたハンドル・リストを含むにはDOS/Vのためのメモリーを解放しなければなりません。SET BLOCK(4AH)ファンクション・コールを使用することにより行うことができます。

68H —
ファイルのコミット

目的

ファイルとして、バッファーに入れられたすべてのデータを装置に書き込みます。この機能はクローズしてオープンする手順の代わりに使用することができます。

例

```
        MOV     AH,68H          ; ファンクション・コール-ファイルの
                                ; 引き渡し
        MOV     BX,Handle       ; ファイルの選択
        INT     21H             ; DOSにリクエストを発行
        JC      Error           ; AX内にエラー・コード

  ----

  Handle        DW      ?       ; 前回のOpenまたはCreateからのハンドル
```

解説

「ファイルのコミット」は、ネットワークのような複数ユーザー環境で、より早くより安全にデータを引き渡しする方法を提供します。

6CH —
拡張オープンおよび作成

目的

任意にファイルをオープンし作成します。

例

```
 MOV     AH,6CH               ; 拡張オープン
 MOV     AL,0                 ; 予約済み
 MOV     BX,MODE              ; オープン・モード
                              ; 形式 : 0WF00000ISSS0AAA
                              ; AAA=アクセス・コード 0=読み取り
                              ;                      1=書き込み
                              ;                      2=読み書き
                              ; SSS=共用モード       0=互換
                              ;                      1=読み書き拒否
                              ;                      2=書き込み拒否
                              ;                      3=読み取り拒否
                              ;                      4=非拒否
                              ; I 0=子プロセスにハンドルを渡す、1=継承なし
                              ; F 0=INT 24H, 1=リターン・エラー
                              ;  このオープンとこのハンドルに対する
                              ;  入出力のエラー・コード
                              ; W 0=引き渡しなし、1=書き込みで自動引き渡し
 MOV     CX,ATTR              ; 属性作成(オープンなら無視)
 MOV     DX,FLAG              ; 機能制御、形式=0000000NNNNEEEE
                              ; NNNN=存在しない動作
                              ;  0=失敗、1=作成
                              ; EEEE=存在する動作
                              ;  0=失敗、1=オープン、2=置き換え/オープン

 MOV     SI,SEG file_name     ; オープンまたは作成する名前
 MOV     DS,SI
 MOV     SI,OFFSET file_name
 INT     21H
 JC      ERROR
                              ; AX=ハンドル
                              ; CX=動作のとられたコード
                              ; 1=ファイルはオープンした
                              ; 2=ファイルはオープンし作成された
                              ; 3=ファイルはオープンし置き換えられた
----
Mode     DW        ?          ; オープン・モード・ビット定義
Attr     DW        ?          ; ファイル属性
Flag     DW        ?          ; 機能定義
File_name    64 DUP (0)
```

解説

ファンクション・コール6CHはファイルのオープン、作成、新規作成で現在利用できる機能を結合します。

Ｆが１の場合、重大エラー・ハンドラー(割り込み24H)は、「拡張オープン」によって返されるハンドルのために無効にされます。このハンドルに発行されたいずれのI/Oも重大エラーは発生しませんが、拡張エラーだけを発生します。

Ｆが０の場合、何の動作も起こりません。

Wが１の場合、「拡張オープン」で返されるハンドルを使用する、どのディスク書き込みも、引き継ぎコール(割込み21H(AL=68H)を参照)を伴います。

Wが０の場合、何の動作も起こりません。

付録C. 装置の入出力制御(IOCtl)

目的

オープンされている装置ハンドルに関連した装置情報の設定、取得、または装置ハンドルに制御文字列を送ったり、装置ハンドルから制御文字列を受け取ります。

解説

次の機能値がAL内で可能です。

AL = 00H 装置情報の取得(DXで返される)
AL = 01H 装置情報の設定(DXで限定)、このコールのためにDHは0でなければなりません。
AL = 02H キャラクター装置からの読み取り
AL = 03H キャラクター装置への書き込み
AL = 04H ブロック装置からの読み取り
AL = 05H ブロック装置への書き込み
AL = 06H 入力状況の取得
AL = 07H 出力状況の取得
AL = 08H 特定ブロック装置が取り外し可能かどうかを限定
AL = 09H 論理装置がローカルであるか、リモートであるかを限定
AL = 0AH ハンドルがローカルであるか、リモートであるかを限定
AL = 0BH 共用再試行回数の変更
AL = 0CH ハンドルの一般IOCtlリクエストの発行
AL = 0DH ブロック装置一般IOCtlリクエストの発行
AL = 0EH 論理ドライブの取得
AL = 0FH 論理ドライブ設定
AL = 10H IOCtlハンドルの問合せ
AL = 11H IOCtlデバイスの問合せ

IOCtlは装置についての情報を得るために使用されます。通常のファイルへのコールをすることができますが、その場合、機能値の00H、06H、07Hだけが定義されています。他のすべてのコールは「無効な機能」エラーを返します。

機能値00Hから08Hは、ネットワーク装置においてサポートされていません。機能値0BHは、ファイル共用コマンドのロード(SHARE)が必要です。

多くのファンクション・コールは、操作が成功するとキャリー・フラグをクリアして返します。エラー状態に出会った場合、キャリー・フラグが設定され、AXは拡張エラー・コードになります。(説明はB-6ページの『拡張エラー・コード』を参照してください。)コール440CHのエラー・コードの説明は、この章のC-15ページにあります。エラー・クラス、位置、推奨される対処方法のようなエラーについての情報は、ファンクション・コール59H (拡張エラーの取得)を発行することによって得られます。

44H —
装置の入出力制御(IOCtl)

AL=00HおよびAL=01Hコール

目的

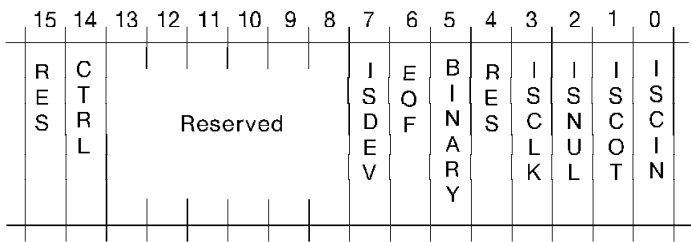
装置情報の設定または取得を行います。

例

```
; 装置情報の取得

        MOV     AH,44H          ; ファンクション・コール-IOCtl
        MOV     AL,0            ; 装置情報の取得を示す
        MOV     BX,Handle       ; 装置の選択
        INT     21H             ; DOSにリクエストを発行
        JC      Error           ; AX内にエラー・コード
        MOV     DevInfo,DX      ; 装置情報の保存

; 装置情報の設定

        MOV     AH,44H          ; ファンクション・コール-IOCtl
        MOV     AL,1            ; 装置情報の設定を示す
        MOV     BX,Handle       ; 装置の選択
        MOV     DX,DevInfo      ; 設定する装置情報
        XOR     DH,DH           ; すべてのDHビットはoffでなければならない
        INT     21H             ; DOSにリクエストを発行
        JC      Error           ; AX内にエラー・コード

-----

DevInfo         DW      ?       ; 装置情報
Handle          DW      ?       ; オープン装置ハンドル
```

解説

DevInfoのビットは次のように定義されます。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES	CTRL	Reserved						ISDEV	EOF	BINARY	RES	ISCLK	ISNUL	ISCOT	ISCIN

ISDEV ＝１ このチャンネルが装置の場合

＝０ このチャンネルがディスク・ファイルの場合(この場合、ビット８から15は０)

DXのビット８から15は、デバイス・ドライバー属性ワードの上位８ビットに相当します。

ISDEV＝１の場合

EOF ＝０ 入力でファイルの終わり(EOF)の場合

BINARY＝１ バイナリー・モードで操作の場合(Ctrl-Z のチェック無し)

BINARY＝０ アスキー・モードで操作の場合(ファイルの終わり(EOF)としてCtrl-Zのチェック)

ISCLK ＝１ この装置がクロック装置の場合

ISNUL ＝１ この装置がNUL装置の場合

ISCOT ＝１ この装置がコンソール出力の場合

ISCIN ＝１ この装置がコンソール入力の場合

CTRL ＝０ この装置が、コールAL=02H、AL=03H、AL=04H、AL=05Hを経て制御文字列を処理できない場合

CTRL ＝１ この装置が、コールAL=02H、AL=03Hを経て制御文字列を処理できれ場合このビットは、ファンクション・コール44Hによって設定できないことに注意してください。

ISDEV＝０の場合

EOF=チャンネルが書き込まれた場合０。ビット０から５はチャンネル(0=A、1=B、．．．)のブロック装置番号です。ビット15、８から13、４は予約されていて変更できません。

注: DHはコールAL=01Hのときには０でなければなりません。

AL=02HおよびAL=03Hコール

目的

これら２つのコールで、キャラクター装置へ制御文字列を送ったり、キャラクター装置から受け取ったりすることができます。

例

```
; キャラクター装置からの制御文字列の読み取り

     MOV     AH,44H           ; ファンクション・コール-IOCtl
     MOV     AL,2             ; IOCtl 読み取りを示す
     MOV     BX,Handle        ; 装置の選択
     MOV     CX,SIZE Buffer   ; 読み取りサイズの設定
     MOV     DI,SEG Buffer    ; I/Oバッファーをアドレス
     MOV     DS,DI
     MOV     DX,OFFSET Buffer ; DS:DXはI/Oバッファーを指す
     INT     21H              ; DOSにリクエストを発行
     JC      Error            ; AX内のエラー・コード
     MOV     Count,AX         ; データ読み取り数の保存

; キャラクター装置への制御文字列の書き込み

     MOV     AH,44H           ; ファンクション・コール-IOCtl
     MOV     AL,3             ; IOCtl書き込みを示す
     MOV     BX,Handle        ; 装置の選択
     MOV     CX,SIZE Buffer   ; 書き込むサイズの設定
     MOV     DI,SEG Buffer    ; I/Oバッファーへアドレス
     MOV     DS,DI
     MOV     DX,OFFSET Buffer ; DS:DXはI/Oバッファーを指す
     INT     21H              ; DOSにリクエストを発行
     JC      Error            ; AX内にエラー・コード
     MOV     Count,A X        ; データが書き込まれた数の保存

-----

Handle  DW   ?                ; オープン・デハイスへのハンドル
Buffer  DB   N DUP(?)         ; I/Oバッファー
Count   DW   ?                ; 実際のI/Oデータ転送数
```

解説

これらは、キャラクター装置の読み取りおよび書き込みコールです。CTRLビットが０の場合、「無効の機能」エラーが返されます。

AL=04HおよびAL=05Hコール

目的

これら２つのコールでブロック装置へ制御文字列を送ったり、ブロック装置から受け取ったりすることができます。

例

```
;   ブロック装置から制御文字列を読み取る

      MOV     AH,44H           ; ファンクション・コール-IOCtl
      MOV     AL,4             ; IOCtl読み取りを示す
      MOV     BL,Drive         ; ドライブの選択
      MOV     CX,SIZE Buffer   ; 読み取りサイズの設定
      MOV     DI,SEG Buffer    ; I/Oバッファーへアドレス
      MOV     DS,DI
      MOV     DX,OFFSET Buffer ; DS:DXはI/Oバッファーを指す
      INT     21H              ; DOSにリクエストを発行
      JC      Error            ; AX内にエラー・コード
      MOV     Count,AX         ; データの読み取られた数を保存

; ブロック装置に制御文字列を書き込む

      MOV     AH,44H           ; ファンクション・コール-IOCtl
      MOV     AL,5             ; IOCtl書き込みを示す
      MOV     BL,Drive         ; ドライブの選択
      MOV     CX,SIZE Buffer   ; 書き込むサイズの設定
      MOV     DI,SEG Buffer    ; I/Oバッファーへアドレス
      MOV     DS,DI
      MOV     DX,OFFSET Buffer ; DS:DXはI/Oバッファーを指す
      INT     21H              ; DOSにリクエストを発行
      JC      Error            ; AX内にエラー・コード
      MOV     Count,AX         ; データの書き込まれた数を保存

-----

Drive         DB   ?           ; ドライブ(0=現行、1=A:, 2=B:, ...)
Buffer        DB   N DUP(?)    ; I/Oバッファー
Count         DW   ?           ; 実際のI/Oデータ転送数
```

解説

これらは、ブロック装置の読み取りおよび書き込みコールです。ドライブ番号は、これらのコールのためにBLにあります。CTRLビットが０の場合「無効な機能」エラーが返されます。ドライブが無効の場合「アクセス拒否」コードが返されます。

AL=06HおよびAL=07Hコール

目的

これらのコールは、ハンドルが入出力に対して準備できているかをチェックできます。

例

```
; 入力装置状況の取得

      MOV     AH,44H          ; ファンクション・コール-IOCtl
      MOV     AL,6            ; IOCtl入力状況を示す
      MOV     BX,Handle       ; 装置の選択
      INT     21H             ; DOSにリクエストを発行
      JC      Error           ; AX内にエラー・コード
      MOV     Status,AL       ; 状況の保存

; 出力装置状況の取得

      MOV     AH,44H          ; ファンクション・コール-IOCtl
      MOV     AL,7            ; IOCtl 出力状況を示す
      MOV     BX,Handle       ; 装置の選択
      INT     21H             ; DOSにリクエストを発行
      JC      Error           ; AX内にエラー・コード
      MOV     Status,AL       ; 状況の保存

-----

Handle        DW      ?       ; 装置をオープンするハンドル
Status        DB      ?       ; 状況us
                              ; ファイルでは：
                              ;   00H = ファイルの終わり
                              ;   FFH = ファイルの終わりではない
                              ; 装置では：
                              ;   00H = 準備中
                              ;   FFH = 準備完了
```

解説

ファイルのために使用される場合、ファイルの終わり(EOF)に達するまでALは常にF2Hを返します。そして、コール42Hで現在のファイル位置が変更されるまで、いつも00Hを返します。装置に対して使用される場合、ALは準備ができている場合はFFH、できていない場合は0が返ります。

AL=08Hコール

目的

このコールにより、装置が取り外し可能なメディアをサポートできるかどうかを限定できます。

例

```
        MOV     AH,44H          ; ファンクション・コール-IOCtl
        MOV     AL,8            ; IOCtl が取り除き可能なことを示す
        MOV     BL,Drive        ; ドライブの選択
        INT     21H             ; DOSにリクエストを発行
        JC      Error           ; AX内にエラー・コード
        MOV     Dtype,AX        ; タイプの保存

  -----

  Drive         DB      ?       ; ドライブ (0=現行、1=A:, 2=B:, ...)
  Dtype         DW      ?       ; ドライブ・タイプ
                                ;   0 = ドライブは取り除き可能
                                ;   1 = ドライブは固定
                                ; 0FH = ドライブは無効
```

解説

AXで返される値が０の場合、装置は取り外し可能です。値が１の場合、装置は固定です。ドライブ番号はBLに入れなくてはなりません。BLの値が無効の場合、「アクセス拒否」が返されます。ネットワーク装置に対しては、「無効な機能」エラーが返されます。

AL=09Hコール

目的

このコールで、論理装置がネットワーク・ディレクトリーと関連づけられているかどうかを限定できます。

例

```
        MOV     AH,44H          ; ファンクション・コール-IOCtl
        MOV     AL,9            ; IOCtlがリモート・ドライブであることを示す
        MOV     BL,Drive        ; ドライブの選択
        INT     21H             ; DOSにリクエストを発行
        JC      Error           ; AX内にエラー・コード
        TEST    DX,1000H        ; ローカルまたはリモートなら参照
        JNZ     Is_Remote       ; ドライブはリモート

  -----

  Drive         DB      ?       ; ドライブ(0=現行、1=A:, 2=B:, ...)
```

解説

エントリーで、BLはチェックしたいブロック装置のドライブ番号(0=省略時値、1=A、2=Bなど)です。DXで返される値は、装置がローカルかリモートかを示します。ビット12は、リモート装置に対して設定されます(1000H)。ビット12は、ローカル装置に対しては設定されません。DXの他のビットは予約されています。ディスク・リダイレクションが一時停止している場合、その機能はビット12を設定せずに返します。

AL=0AHコール

目的

このコールでローカル装置のハンドルなのか、またはネットワークと繋がるリモート装置のハンドルなのかを限定できます。

例

```
        MOV     AH,44H          ; ファンクション・コール-IOCtl
        MOV     AL,0AH          ; IOCtlはリモート・ハンドルであることを示す
        MOV     BX,Handle       ; 装置およびファィルの選択
        INT     21H             ; DOSにリクエストを発行
        JC      Error           ; AX内にエラー・コード
        TEST    DX,8000H        ; ローカルまたはリモートなら参照
        JNZ     Is_Remote       ; ドライブはリモート

-----

Handle          DW      ?       ; ファイルまたは装置をオープンするハンドル
```

解説

リモート装置の場合、ビット15が設定されます(8000H)。ハンドルはBX内に置かれなければなりません。ビット15はローカル装置に対しては設定されません。

AL=0BHコール

目的

共用と資源の衝突のロックに対する再試行を制御します。

例

```
        MOV     AH,44H          ; ファンクション・コール - IOCtl
        MOV     AL,0BH          ; IOCtlの再試行数の設定を示す
        MOV     CX,NumLoops     ; ループ数の設定
        MOV     DX,NumRetries   ; 再試行回数の設定
        INT     21H             ; DOSにリクエストを発行
        JC      Error           ; AX内にエラー・コード

  -----

  NumLoops      DW      ?       ; 上記のループを実行する回数
  NumRetries    DW      ?       ; エラーのときに再試行する回数
```

解説

すべての共用とロックの衝突は、それらがDOS/Vエラーまたは重大エラーとして返される前に、自動的に何回も再試行されます。再試行の回数と再試行の遅延時間を選択できます。入力に際しては、CXは遅延ループを実行する回数です。DXは再試行数です。遅延ループは次の順序で構成されています。

```
XOR     CX,CX
LOOP    $           ; 64K回ループする
```

このコールが発行されない場合、DOS/VはCXおよびDXの省略時値として遅延=1、再試行=3を使用します。アプリケーション・プログラムに短時間だけの共用またはロックの衝突を起こすことが予想される場合、実際にアプリケーション・プログラムに返されるエラー数を最小にするために、CXおよびDXの値を大きくすることができます。

AL=0CHコール

目的

この一般IOCtl機能は、オープン装置ハンドルを使用って、コード・ページ切り替えや装置情報の取得と設定を、デバイス・ドライバーにリクエストします。

注: この機能は英語モードのみでサポートされています。

例

```
MOV     AH,44H           ; ファンクション・コール-IOCtl
MOV     AL,0CH           ; ファイル・ハンドル一般IOCtlリクエストを示す
MOV     BX,Handle        ; 装置およびファィルの選択
MOV     CH,Category      ; 装置タイプの選択
MOV     CL,Function      ; 機能の設定
MOV     DI,SEG Packet    ; サブファンクション・パラメーター・
                         ; パケットへアドレス
MOV     DS,DI
MOV     DX,OFFSET Packet ; DS:DXパラメーター・パケットを指す
INT     21H              ; DOSにリクエストを発行
JC      Error            ; AX内にエラー・コード

-----

Handle        DW      ?        ; ファイルまたは装置をオープンするハンドル
Category      DB      ?        ; 装置タイプ
                               ; 0 - 未定義(装置タイプが未定義の場合)
                               ; 1 - COMx装置
                               ; 3 - CON
                               ; 5 - LPTx装置
Function      DB      ?        ; 範囲内の機能
                               ; 範囲3と5のため：
                               ;   4CH = 開始準備
                               ;   4DH = 終了準備
                               ;   4AH = コード・ページの選択(設定)
                               ;   6AH = 選択したコード・ページの
                               ;         問い合わせ(取得)
                               ;   6BH = 準備リストの問い合わせ
                               ; 範囲3のために：
                               ;   5FH = ディスプレイ情報の設定
                               ;   7FH = ディスプレイ情報の取得
```

開始準備: CL=4CHの時、DS:DXによって指されるパラメーター・ブロックは、次のようなレイアウトです。

```
Packet          Label               Word
  PS_PACKET     STRUC             ; 準備開始パケット
  PS_FLAGS      DW      0         ; 制御フラグ
                                  ; ビット 0=0: ダウンロード準備
                                  ; ビット 0=1: カートリッジ準備
                                  ; 他は予約済み(0に設定)
  PS_LENGTH     DW      (n+1)*2   ; 残りのパケットの長さのバイト数
  PS_NUMCP      DW      n         ; コード・ページ数
  PS_CP1        DW      ?         ; コード・ページ1
                    .
                    .
                    .
                    .
  PS_CPn        DW      ?         ; コード・ページn
  PS_PACKET     ENDS
```

注:

1. PS_CPnから-1までの設定は、デバイス・ドライバーにその位置のコード・ページ値を変更しないよう命じます。他の値はすべてあるコード・ページが準備されます。

2. nはCONFIG.SYS内のDEVICE=コマンドで指定される、追加コード・ページ数です。nの値は12までです。

3. カートリッジ準備のためにPS _FLAGSフィールドを1に設定します。

「開始準備」リクエストは、コード・ページの準備を始めます。その後にデバイス・ドライバーへ、コード・ページ・フォントを定義している書き込みデータが続きます。これには1つ以上のIOCtl書き込み制御文字列コール(AX=4403H)を使用します。この情報は装置へダウンロードされるものと想定されています。ストリームは「終了準備」によって終わります。ストリームの形式は装置に依存します。

情報が(システムの故障または電源オフによって)失われた場合、準備されたコード・ページを再書き込みする必要はありません。すべてのコード・ページ値(PS_CPn)を-1に設定して、デバイス・ドライバーに「開始準備」リクエストを発行することによって「リフレッシュ」操作をリクエストすると、最も新しく準備されたコード・ページ情報が装置に復元されます。この操作に「終了準備」リクエストをすぐに続けなければなりません。データが準備操作のために書き込まれない場合、ドライバーは新しく準備されたコード・ページをハードウェア・コード・ページと解釈します。これで、ユーザー変更可能ハードウェア・フォント(通常、カートリッジ内にある)をサポートする装置が可能になります。

ハードウェア定義コード・ページのための準備は必要ありません。

開始準備エラー・コード

コード	意味
01	無効な機能番号
22	未定義コマンド
27	コード・ページ矛盾(KEYBxx不一致に使用される)
29	装置エラー
31	デバイス・ドライバーは装置にダウンロードするコード・ページのコピーを持ちません。

書き込みエラー・コード

コード	意味
27	装置がファイル内に見つからない、またはコード・ページがファイル内に見つからない。
29	装置エラー
31	ファイル内容がフォント・ファイルでない、またはファイルの内容が損傷を受けている。

終了準備:　CL=4DHの時、DS:DXで指されるパラメーター・ブロックは、次のようなレイアウトです。

```
Packet          Label        Word
  PE_PACKET     STRUC             ; 終了準備パケットt
  PE_LENGTH     DW      2         ; パケット長のバイト値ytes
  PE_RESV1      DW      0         ; (予約済み、0でなければならない)
  PE_PACKET     ENDS
```

終了準備エラー・コード

コード	意味
19	フォント・ファイルから間違ったデータの取得
31	開始準備なし

選択コード・ページの選択／問合せ:　CL=4AHまたは6AHの時、DS:DXで指される
パラメーター・ブロックは、次のようなレイアウトです。

```
PACKET LABEL WORD
 CP_PACKET     STRUC                  ; 選択／問合せ選択パケット
 CP_LENGTH     DW      2+(n+1)*2      ; パケット長のバイト値
 CP_CPID       DW      ?              ; コード・ページID
 CP_VECTOR1    DB      ?,?            ; DBCSベクトル１
                               .
                               .
                               .
CP_VECTORn     DB      ?,?            ; DBCSベクトルｎ
               DB      0,0            ; 終了マーカー
CP_PACKET      ENDS
```

「選択／問合せ選択コード・ページ」もDBCS環境ベクトルを含みます。デバイス・ドライバーによってはコード・ページ値のみをサポートすることもあります。結果として、DBCS情報があるかどうかを決定するために、このコールを使用する場合、返される長さをチェックしなければなりません。DOS/Vのアジア・バージョンに添えて提供されるドライバーのみが、これをサポートします。

コード・ページ選択エラー・コード

コード	意味
26	コード・ページが準備されていない
27	現行キーボードは、このコード・ページをサポートしていない
29	装置エラー

選択されたコード・ページ問合せエラー・コード

コード	意味
26	コード・ページが選択されていない
27	装置エラー

準備リストの問合せ:　CL=6BHの時、DS:DXによって指されるパラメーター・ブロックは、次のようなレイアウトです。

```
PACKET              LABEL              WORD
  QL_PACKET      STRUC                 ; 問合せリスト・パケット
  QL_LENGTH      DW((m+1)+(n+1))*2     ; パケット長のバイト値
  QL_NUMHWCP     DW       n            ; ハードウェア・コード・ページ数
  QL_HWCP1       DW       ?            ; ハードウェア・コード・ページ1
                     .
                     .
                     .
                     .
  QL_HWCPn       DW       ?            ; ハードウェア・コード・ページ n
  QL_NUMCP       DW       n            ; 準備されたコード・ページ数
  QL_CP1         DW       ?            ; 準備されたコード・ページ1
                     .
                     .
                     .
                     .
  QL_CPm         DW       ?            ; 準備されたコード・ページn
  QL_PACKET      ENDS
```

注:　デバイス・ドライバーは、(ハードウェアまたは準備された)コード・ページの各タイプに対して12までのコード・ページ値を返せます。従ってn、mともに12までの値を取れます。

準備リスト問合せエラー・コード

コード	意味
26	コード・ページが選択されていない
29	装置エラー

ディスプレイ情報の取得／設定:　CL=5FHまたは7FHの時、DS:DXによって指される パラメーター・ブロックは、次のようなレイアウトです。

```
VP_PACKET    STRUC           ; ビデオ・パラメーター・パケット
VP_LEVEL     DB      0       ; リクエストされた情報レベル
VP_RESV1     DB      0       ; (予約済み、0でなければばならない)
VP_LENGTH    DW      14      ; 残りのパケット長のバイト値
                             ; ビット0=0: 強調カラー
                             ; ビット0=1: ブリンク
                             ; 他予約済み (0に設定)
VP_MODE      DB      ?       ; ビデオ・モード
                             ; 1 = テキスト
                             ; 2 = APA
                             ; (予約済み. 0でなければばならない)
VP_RESV2     DB      0       ; カラー数 (白黒=0)
VP_COLORS    DW      ?       ; 表示幅のピクセル数(APAモードのみ)
VP_WIDTH     DW      ?       ; 表示長さのピクセル数(APAモードのみ)
VP_LENGTH    DW      ?       ; 表示長さの文字数
VP_COLS      DW      ?       ; 表示幅の文字数
VP_ROWS      DW      ?       ; 表示長さの文字数

VP_PACKET    ENDS
```

AL=0DHコール

目的

この一般IOCtl機能は、次の中のひとつの副機能を実行することをブロックデバイス・ドライバーにリクエストします。

- 装置パラメーターの取得
- 装置パラメーター設定
- 論理装置のトラック読み取り
- 論理装置のトラック書き込み
- 論理装置のトラックの形式とベリファイ
- 論理装置のトラックのベリファイ
- アクセス・フラグ状況の取得
- アクセス・フラグ状況の設定

例

```
        MOV     AH,44H           ; ファンクション・コール-IOCtl
        MOV     AL,0DH           ; ブロック装置一般IOCtlを示す
        MOV     BL,Drive         ; 装置／ファイルの選択
        MOV     CH,Category      ; 装置タイプの設定
        MOV     CL,Function      ; 機能の設定
        MOV     DS,SEG Packet    ; 副機能パラメーター・パケット・アドレス
        MOV     DX,OFFSET Packet
        INT     21H              ; DOSにリクエストを発行
        JC      Error            ; AX内にエラー・コード

  -----

  Drive         DB      ?        ; ドライブ(0=現行、1=A:, 2=B:, ...)
  Category      DB      ?        ; 装置のタイプ
                                 ; 8-ブロック装置
  Function      DB      ?        ; カテゴリー中の機能
                                 ; カテゴリー 8のため：
                                 ;   40H = 装置パラメーターの設定
                                 ;   60H = 装置パラメーターの取得
                                 ;   41H = 論理装置のトラック書き込み
                                 ;   61H = 論理装置のトラック読み取り
                                 ;   42H = 論理装置のトラック・フォーマットと
                                 ;         ベリファイ
                                 ;   62H = 論理装置のトラック・ベリファイ
                                 ;   47H = アクセス・フラグの設定
                                 ;   67H = アクセス・フラグの取得
                                 ;   68H = 媒体タイプの取得
```

注: 機能43Hから46Hと、63Hから66Hはシステムのために予約されています。

解説

CHはメジャー・コード(すべての機能にたいして08H)で、CLはマイナー・コード(機能)です。

装置パラメーターの取得または設定

装置パラメーターを取得するには、CL=60Hに設定してください。

装置パラメーターを設定するには、CL=40Hに設定してください。

CL=60HまたはCL=40Hの時、パラメーター・ブロックは次のようなフィールドレイアウトです。

```
Packet                  Label       Byte
 A_deviceParameters     STRUC
 SpecialFunctions       DB      ?
 DeviceType             DB      ?
 DeviceAttributes       DW      ?
 NumberOfCylinders      DW      ?
 MediaType              DB      ?
 DeviceBPB              a_BPB   <>
 TrackLayout            a_TrackLayout <>
 A_deviceParameters     ends
```

パラメーター・ブロックの各フィールドの説明を次に示します。

SpecialFunctionsフィールド

この１バイト・フィールドは、「装置パラメーターの取得および設定」機能をさらに定義するために使用されます。

「装置パラメーターの取得」機能に対して、「Special Functions」フィールドのビット０は次のような意味です。

ビット0=1 BUILD BPBが返すBPBを返します。
ビット0=0 装置の省略時BPBを返します。

注: 他のすべてのビットはオフでなければなりません。

「装置パラメーター設定」のために**Special Functions**フィールドのビット０、１、２が使用されます。

これらのビットは、CL=40Hの時に次の意味を持ちます。

ビット0=1 この後すべてのBUILD BPBリクエストは、**DeviceBPB**を返します。ビット０がリセットされた「デバイス設定」リクエストが受け取られると、BUILD BPBは、実際のメディアBPBを返します。

ビット0=0 **DeviceBPB**フィールドが、この装置のために新しい省略時BPBを含むことを示します。以前の「装置設定」リクエストがこのビットをオンに設定した場合、実際のメディアBPBが返されます。そうでなければ、装置の省略時BPBは、BUILD BPBによって返されます。

ビット1=1 **Track Layout**フィールドを除くパラメーター・ブロックのすべてのフィールドを無視します。

ビット1=0 パラメーター・ブロックのすべてのフィールドを読み取ります。

ビット2=1 トラック内のすべてのセクターが同じサイズで、すべてのセクター数が1とnの間にあることを示します(ここでnはトラック内のセクター数です)。

ビット2=0 トラック内のすべてのセクターが同じサイズではない場合があることを示します。

注:

1. すべての他のビットはリセットされなければなりません。
2. 通常のトラック・レイアウトのためにビット2を設定します。ビット2が設定されると「フォーマット・トラック」はより効果的となります。
3. 同時にビット0と1を設定することは無効で、エラーとみなされます。

DeviceTypeフィールド

この1バイト・フィールドは、物理装置タイプを記述します。装置タイプはIOCtlによっては設定されず、装置から受け取られます。

このフィールドの値は次の意味があります。

0 = 320/360KB, 5.25インチ
1 = 5.25インチ、1.2MB
2 = 3.5インチ、720KB
3 = 8インチ単密度
4 = 8インチ倍密度
5 = 固定ディスク
6 = テープ・ドライブ
7 = その他

DeviceAttributesフィールド

装置の物理属性を記述する1ワード・フィールドです。装置属性はIOCtlによって設定されず、デバイス・ドライバーから受け取られます。

このフィールドのビット0と1のみが使用されます。それらは次の意味があります。

ビット0=1 メディアは取り外し可能です。
ビット0=0 メディアは取り外し可能です。
ビット1=1 ディスケット・チェンジラインがサポートされます。
ビット1=0 ディスケット・チェンジラインはサポートされません。

ビット２～15は予約されています。

NumberOfCylindersフィールド

このフィールドは、メディア・タイプとは独立した物理装置にサポートされた、シリンダーの最大数を示します。このフィールドの情報はIOCtlによって設定されず、デバイス・ドライバーから受け取られます。

MediaTypeフィールド

マルチ・メディア・ドライブに対して、このフィールドはどのメディアがドライブにあるかを示します。このフィールドは「装置パラメーターの設定」(CL=40H)副機能に対してのみ意味があります。

Media Typeフィールドは、ドライブで実際のメディアが決定できない時のみ使用されます。メディア・タイプは装置タイプに依存します。

装置タイプに関係なく、０の値は省略時値を表します。たとえば、5.25インチ1.2MBディスケット・ドライブはマルチ・メディア・ドライブです。メディア・タイプは次のように定義されます。

0=4倍密度1.2MB(96tpi)ディスケット

1=倍密度320/360KB(48tpi)ディスケット

1.2MBドライブの省略時メディア・タイプは、４倍密度1.2MBディスケットです。

DeviceBPBフィールド

「装置パラメーターの取得」機能について:

- **Special Functions**フィールドのビット０が設定されている場合、デバイス・ドライバーは、BUILD BPBが返すBPBを返します。
- **Special Functions**フィールドのビット０が設定されていない場合、デバイス・ドライバーは、装置のために省略時BPBで返します。

「装置パラメータ設定」機能について:

- **Special Functions**フィールドのビット０が設定されている場合、「デバイス・パラメーター設定」リクエストが「Special Functions」フィールド・リセットのビット０が　受け取られるまで、その後のすべてのBUILD BPBＢリクエストに対して、デバイス・ドライバーはこのフィールドからBPBを返すことをリクエストされます。
- ビット０が設定されていない場合、このフィールドに含まれるBPBはデバイスのための新しい省略時BPBとなります。

DeviceBPBフィールドは、次のような形式です。

```
a_BPB               STRUC
BytesPerSector      DW      ?
SectorsPerCluster   DB      ?
ReservedSectors     DW      ?
NumberOfFATs        DB      ?
RootEntries         DW      ?
TotalSectors        DW      ?
MediaDescriptor     DB      ?
SectorsPerFAT       DW      ?
;
SectorsPerTrack     DW      ?
Heads               DW      ?
HiddenSectors       DD      ?
BigTotalSectors     DD      ?
Reserved            DB      6 Dup (0)
a_BPB               ENDS
```

TrackLayoutフィールド

これは、メディア・トラック上の考えられるセクター・レイアウトを示す可変長テーブルです。

DOS/Vデバイス・ドライバーは、各論理装置のトラック・レイアウト・テーブルを保持しません。グローバル・トラック・テーブルはメディアの属性が変わる時(「装置パラメーターの設定」副機能によって)更新されなければなりません。

注: 「装置パラメーターの設定」副機能 (CL=40H)は、**SpecialFunctions**フィールドのビット１がどのように設定されているかに関係なく、トラック・テーブルを修正します。

「装置パラメーターの取得」ではこのフィールドは使用されません。トラック・レイアウトはその後の「トラック読み取り／書き込み」、「トラック・フォーマット／ベリファイ」および「トラック・ベリファイ」機能によって使用されます。

次の例はこのフィールドがどのように形式されるかを示しています。

```
Total sectors-------|SectorCount      DW  n

Sector  1-----------|SectorNumber_1   DW  1H
                    |SectorSize_1     DW  200H

Sector  2-----------|SectorNumber_2   DW  2H
                    |SectorSize_2     DW  200H

Sector  3-----------|SectorNumber_3   DW  3H
                    |SectorSize_3     DW  200H

Sector  4-----------|SectorNumber_4   DW  4H
                    |SectorSize_4     DW  200H
                            |
                            |
                            |
Sector  n-----------|SectorNumber_n   DW  n
                    |SectorSize_n     DW  200H
```

注: すべての値は16進数です。

セクターの全数は**SectorCount**フィールドによって示されます。各セクター番号は固有であり、１からｎ(セクター・カウント)の範囲になければばなりません。上の例で示されるように、最初のセクター番号は１で、最後のセクター番号はセクター・カウント(ｎ)と同じです。**SpecialFunctions**フィールドのビット２が設定される場合、バイトで測定されるすべてのセクター・サイズは同じでなければなりません。**SpecialFunctions**フィールドのビット２の記述を参照してください。

注: **DeviceType**、**DeviceAttributs**および**NumberOfSectors**フィールドは、物理デバイスが変更された場合にのみ変更すべきです。

論理装置のトラック読み取り／書き込み

論理装置のトラックを読み取るにはCL=61Hに設定してください。

論理装置のトラックに書き込むにはCL=41Hに設定してください。

パラメーター・ブロックは、論理装置のトラックに読み取り／書き込みをする時、次のようなレイアウトです。

```
Packet                  LABEL   BYTE
 a_ReadWriteTrackPacket  STRUC
 SpecialFunctions        DB      ?
 Head                    DW      ?
 Cylinder                DW      ?
 FirstSector             DW      ?
 NumberOfSectors         DW      ?
 TransferAddress         DD      ?
 A_ReadWriteTrackPacket  ENDS
```

注:

1. **SpecialFunctions**フィールドのすべてのビットはリセットされなければなりません。

2. **FirstSector**フィールドと**NumberOfCylinders**フィールドの値は０基準です。たとえばセクター９を示すには値は８に設定します。

論理装置のトラックのフォーマット／ベリファイ(IOCtl書き込み)

トラックをフォーマットしベリファイするには、CL=42Hに設定してください。

トラックをベリファイするには、CL=62Hに設定してください。

パラメーター・ブロックは、論理装置のトラックをフォーマットしベリファイする時に、次のようなレイアウトです。

```
PACKET                    LABEL   BYTE
 A_FormatPacket           STRUC
 SpecialFunctions         DB      ?
 Head                     DW      ?
 Cylinder                 DW      ?
 A_FormatPacket           ENDS
```

エントリーの際、SpecialFunctionsフィールドのビット０は次のような意味があります。

ビット0=1 トラック数とトラック当たりのセクター数の組み合わせがサポートされているかどうかを調べるために、形式ステータスのチェックをコールします。
ビット0=0 トラック・フォーマット／ベリファイ・コール。

「トラック数」と「トラック当たりのセクター数」の組み合わせがサポートされているかどうかを調べるために、「装置パラメーター設定」コールを、その組み合わせに対して正しいBPBで発行しなければなりません。この発行は「フォーマット状況」コールを発行する前に行ってください。デバイス・ドライバーは何がサポートされているかを示す、正しいコードを返すことができます。「フォーマット状況チェック」コールに対して、SpecialFunctionsフィールド内に返される値は次のとおりです。

0= この機能はROM BIOSによってサポートされています。トラック数とトラック当たりのセクター数の指定された組み合わせは、ディスケット・ドライブに対して許可されています。
1= この機能は、ROM BIOSによってサポートされていません。
2= この機能は、ROM BIOSによってサポートされています。トラック数とトラック当たりのセクター数の指定された組み合わせは、ディスケット・ドライブに対して許可されていません。
3= この機能は、ROM BIOSによってサポートされていますが、ディスケット・ドライブが空であるために、ROM BIOSはトラック数とトラック当たりのセクター数が許可されているかどうかを限定できません。

トラックを形式するためには

1. 「装置パラメーターの設定」ファンクション・コールを発行してください。

2. トラック数とトラック当たりのセクター数の組み合わせを有効にするために、「フォーマット状況チェック」ファンクション・コールを発行してください。値として１が返された場合、ROM BIOSがこの機能をサポートしていないためその結果を無視してください。

3. そのメディアの各トラックに対しSpecialFunctionsフィールドのビット０をリセットして、「トラック・フォーマット／ベリファイ」ファンクション・コールを発行してください。

AccessFlag状況の取得／設定

固定ディスクのアクセス・フラグ状況を取得するには、CL=67Hに設定してください。

固定ディスクのアクセス・フラグ状況を設定するには、CL=47Hに設定してください。

パラメーター・ブロックは、固定ディスクのアクセス・フラグ状況を取得または設定する時、次のようなレイアウトです。

```
PACKET                    LABEL           BYTE
a_DiskAccess_Control      STRUC
SpecialFunctions          DB     0
DiskAccess_Flag           DB    ?         ; 0 = ディスク・アクセス不可
                                          ; 他の値 = ディスク・アクセス可
a_DiskAccess_Control      ENDS
```

メディアがフォーマットされていない、または無効なブート・レコードを持っている場合、システムはメディアに対してディスクI/Oを許可しません。これにより、固定ディスクのデータ保全性を確保します。メディアのフォーマットは特殊な動作で、未フォーマット・メディアに対してディスクI/Oを実行する必要がありますから、ディスク・アクセス・フラグを制御するための追加機能が必要です。フォーマット・ユーティリティーは、未フォーマット・メディアにアクセスするためにDiskAccess_Flagが0でない値の場合、「アクセス・フラグ設定(CL=47H)」を発行しなければなりません。各フォーマット操作が成功したら、さらに一般ユーザーからのディスクI/Oを可能にするために、アクセス・フラグ状況はそのままにしておいてください。フォーマットに失敗したら、メディア・アクセスをそれ以上進ませないためにDiskAccess _Flagを０にして「アクセス・フラグ・ステータス設定(CL=47H)」を発行してください。システム・ディスク・アクセス・フラグの現行状況を取得するためには、「アクセス・フラグ・ステータスの取得(CL=67H)」を発行してください。DiskAccess_Flagが０の場合、ディスクI/Oはメディアに対して許可されません。

Media IDの取得／設定

メディア識別コードを取得するには、CL=66Hに設定してください。

メディア識別コードを設定するには、CL=47Hに設定してください。

パラメーター・ブロックは、メディア識別コードを取得または設定する時、次のようなレイアウトです。

```
PACKET              LABEL       BYTE
Media_ID_Packet     STRUC
Info_Level          DB          0               ; 情報レベル
                                                ; 現状ではつねに0
Serial              DD          ?               ; ボリューム通し番号
Label               DB          11 dup (' ')    ; ブート・レコードからの
                                                ; ボリューム・ラベル
File_Sys_Type       DB          8 dup (' ')     ; ファイル・システム・
                                                ; タイプ
Media_ID_Packet     ENDS
```

メディア識別コードの取得は、ブート・レコードからMedia _ID_Packetに情報をコピーします。メディア識別コードの設定は、Media_ID_Packetからブート・レコードに情報をコピーします。ディスクが正しいBPBを持っていない場合、または署名フィールドがない場合は、処理は行われず、両機能は未知のメディア（エラー・コード7）を戻します。

Media Typeの取得（DOS/V）

メディア・タイプを取得するには、CL=68Hに設定してください。

パラメーター・ブロックは、次のようなレイアウトです。

```
PACKET              LABEL     BYTE
Media_Type_Packet   STRUC
Default             DB        ?        ; メディアがドライブ容量と
                                       ; 同じか同等の場合は1。
                                       ; 少ない場合は0。
Media_Type          DB        ?        ; メディア・タイプ
                                       ; 2 : 720KB 3.5inch 80track
                                       ; 7 : 1.44MB 3.5inch 80track
Reserved_1          DB        ?        ; 予約済み
Reserved_2          DD        ?        ; 予約済み
Media_ID_Packet     ENDS      ?
```

解説

キャリーをセットしている場合は、エラーの戻りコードはAXに入ります。

AX=8102Hの場合は、ドライブが作動可能でありません。

AX=8107Hの場合は、メディア・タイプが未定義のものです。ドライブは機能または
メディアをサポートしません。

AL=0EHコール

目的

１つ以上の論理ドライブがブロック装置に割り当てられるかどうかを限定することを、デバイス・ドライバーに許可します。このコールが発行されると、入力の際ドライブ番号はBLで渡されます。

例

```
        MOV     AH,44H          ; ファンクション・コール-IOCtl
        MOV     AL,0EH          ; 論理ドライブのチェックを示す
        MOV     BL,Drive        ; ドライブ選択
        INT     21H             ; DOSにリクエストを発行
        JC      Error           ; AX内にエラー・コード
        CMP     AL,0            ; この装置に対しドライブ文字は1つ?
        JE      Single_Drive    ; はい!
        MOV     ActiveDrive,AL  ; アクティブ・ドライブ情報の保存

  -----

  Drive         DB      ?       ; ドライブ (0=現行、1=A:, 2=B:, ...)
  ActiveDrive   DB      ?       ; この装置に対する現行ドライブ文字
                                ; (1=A:, 2=B:, ...)
```

解説

ブロック装置が、それに割り当てられた１つ以上の論理ドライブ文字を持っている場合、出力の際装置を参照するために使用された、最後のドライブ文字に相当するドライブ番号が、ALで返されます。１つのドライブ文字のみが装置に割り当てられている場合、このコールによって０がALで返されます。

AL=0FHコール

目的

このコールはブロック装置を参照するために使用される、次の論理ドライブ文字を変更することを、デバイス・ドライバーにリクエストします。

例

```
        MOV      AH,44H          ; ファンクション・コール-IOCtl
        MOV      AL,0FH          ; 論理ドライブの設定を示す
        MOV      BL,Drive        ; ドライブ設定
        INT      21H             ; DOSにリクエストを発行
        JC       Error           ; AX内にエラー・コード
        CMP      AL,0            ; この装置に対しドライブ文字は1つだけ?
        JE       Single_Drive    ; はい!
        MOV      ActiveDrive,AL  ; アクティブ・ドライブ情報の保存
                                 ; (BL内と同じでなければならない)

  -----

  Drive          DB      ?       ; ドライブ (0=現行、1=A:, 2=B:, ...)
  ActiveDrive    DB      ?       ; この装置に対する現行ドライブ文字
                                 ; (1=A:, 2=B:, ...)
```

解説

ドライブ上のディスケットをコピーする場合で、物理ドライブ番号に１つ以上の論理ドライブ文字が割り当てられている場合(たとえばシングル・ドライブでのコピー)DOS/Vはディスケットのスワップ・プロンプトを発行します。これにより、どの論理ドライブ文字が、現在物理ドライブ番号を参照しているかが分かります。ドライブがソースからターゲットに変わると、DOS/Vは該当するディスクにメディアを挿入するメッセージを発行します。

AL=0FHコール(論理ドライブの設定)を発行することによって、このメッセージを避けることができます。

DOS/Vのディスケットのスワップ・メッセージを避けるためには、次のI/Oリクエストで参照されるドライブ文字に相当するドライブ番号を、BLに設定してください。

注: AL=0EHコールを発行することにより、物理ドライブ番号に割り当てられた最後の論理ドライブ文字を決定できます。

どのブロック装置も論理ドライブを持つことができるので、このコールは複数のドライブ文字を含む、すべてのI/O操作以前に、発行しなければなりません。そうでなければ、DOS/Vメッセージが発行されることがあります。

AL=10Hコール

目的

問合せIOCtlハンドルはデバイス・ハンドルを指定し、指定した特定のIOCtl機能をデバイスがサポートしているかどうかを判別します。

例

```
MOV     AH,44H          ; ファンクション・コール-IOCtl
MOV     AL,10H          ;
MOV     BX,handle       ; デバイスのハンドル
MOV     CH,category     ; デバイスのカテゴリー
MOV     CL,function     ; チェックするファンクション
                        ;
MOV     DX,offset packet
INT     21H             ; DOSに対するリクエスト
JC      Error           ; AXのなかのエラー・コード
```

解説

カテゴリー、ファンクションおよびパラメーター・ブロックを、チェックされる機能のために指定します。

出口でキャリーがセットされている場合は、ALのなかのエラー・コードは「機能がサポートされていない」を意味する１です。キャリーがセットされていない場合は、AXは０です。

この機能呼出はDOS/Vデバイス・ドライバーによってサポートされます。

AL=11Hコール

目的

問合せIOCtlハンドルはドライブ番号を指定し、指定した特定のIOCtl機能をドライブがサポートしているかどうかを判別します。

例

```
MOV     AH,44H          ; ファンクション・コール-IOCtl
MOV     AL,11H          ;
MOV     BX,drive        ; ドライブ
MOV     CH,category     ; デバイスのカテゴリー
MOV     CL,function     ; チェックするファンクション
                        ;
MOV     DX,offset packet
INT     21H             ; DOSに対するリクエスト
JC      Error           ; AXのなかのエラー・コード
```

解説

カテゴリー、ファンクションおよびパラメーター・ブロックを、チェックされる機能のために指定します。

出口でキャリーがセットされている場合は、ALのなかのエラー・コードは「機能がサポートされていない」を意味する１です。キャリーがセットされていない場合は、AXは０です。

この機能呼出はDOS/Vデバイス・ドライバーによってサポートされます。

付録D. EMSメモリー・サポート

拡張メモリーは、EMSメモリー仕様デバイス・ドライバーとEMS可能ハードウェア・アダプターの組み合わせで、メモリーをアドレス指定できます。

下の表は、Lotus/Intel/Microsoft(LIM)[**]拡張メモリー・マネージャー仕様バージョン4.0の機能を記述したもので、DOS/Vでどれがサポートされているかを示しています。これらのコールに関する詳細な情報とガイドラインについては、「LIM仕様書」を参照してください。

LIM	INT 67H	インターフェース	DOS/Vサポート	意味
1	AH=40H	App	Yes	状況の取得
2	AH=41H	App	Yes	ページ・フレーム・アドレスの取得
3	AH=42H	App	Yes	未割り当てページ数の取得
4	AH=43H	App	Yes	ページ割り当て
5	AH=44H	App	Yes	マップ／アンマップ・ハンドル
6	AH=45H	App	Yes	ページ割り当て解除
7	AH=46H	App	Yes	EMMバージョンの取得
8	AH=47H	App	Yes	ページ・マップ保存
9	AH=48H	App	Yes	ページ・マップ復元
10	予約済み			
11	予約済み			
12	AH=4BH	App	Yes	EMMハンドル・カウントの取得
13	AH=4CH	App	Yes	EMMハンドル・ページの取得
14	AH=4DH	App	Yes	全EMMハンドル・ページの取得
15	AH=4EH	App	Yes	ページ・マップの取得／設定
16	AH=4FH	App	Yes	部分ページ・マップの取得／設定

[**] LotusはLotus社の商標です。IntelはIntel社の商標です。またMicrosoftはMicrosoft社の商標です。

LIM	INT 67H	インターフェース	DOS/Vサポート	意味
17	AH=50H	App	Yes	複数ハンドル・ページ・マップ／アンマップ
18	AH=51H	App	Yes	ページ再割り当て
19	AH=52H	App	No	ハンドル属性の取得／設定
20	AH=53H	App	Yes	ハンドル名の取得／設定
21	AH=54H	App	Yes	ハンドル・ディレクトリーの取得
22	AH=55H	App	Yes	ページ・マップ変更と飛び越し
23	AH=56H	App	Yes	ページ・マップ変更とコール
24	AH=57H	App	Yes	メモリー範囲移動／交換
25	AH=58H	App	Yes	マップ可能物理アドレス配列の取得
26	AH=59H	OS	Yes	拡張メモリー・ハードウェア情報の取得
27	AH=5AH	App	Yes	新ページ割り当て
28	AH=5BH	OS	Yes	副ページ・マップ・レジスター設定
29	AH=5CH	App	Yes	ウォーム・ブートのための拡張メモリー・ハードウェアの準備
30	AH=5DH	App	Yes	OS/E機能の使用可／不可の設定

付録E. タスク・スワッピング

DOS/Vで提供されるDOSシェルには、タスク・スワッパー機能があります。これを使用して、ユーザーは１つの適用業務から別の適用業務に、いずれの適用業務も終了せずに切り替えることができます。ユーザーが新しいプログラムを起動すると、タスク・スワッパーにより、現在活動状態にあるプログラムが保留され、レジスターの内容すべてを保管し、プログラム・メモリーの内容をディスクの“SET TEMP=” で指定されている場所に書き込みます。そして、新しいプログラムをロードし、実行することによって、新しいセッションが作成されます。（セッションとは、タスク・スワッパーにより直接実行され、他のセッションに依存せずに動作するプログラムのことです。）タスク・スワッパーは活動状態のままで、通常、あらかじめ定義されたキー文字列の入力を待ってキーボードを監視します。ユーザーがこのキー文字列を入力すると、タスク・スワッパーは現行セッションを保留および保管し、以前に保留された他のセッションに制御を移します。タスク・スワッパーは保留されていたセッションのメモリーの内容をシステム・メモリーに読み込み、レジスターを保管された値に戻して、そのセッションに制御を移します。

ほとんどの適用業務プログラムは、問題なく保留することができます。それらは同期的に実行されるので、実行の途中で割り込みを受けてスワップされても、再起動できます（タスク・スワッパーが、プログラムを保留したときのシステムの状況を正しく保管し、復元する場合）。プログラムの中には、タスク・スワッパーで安全に動作させることのできないものもあります。たとえば次のような場合です。

- いくつかの終了後常駐(TSR)型のプログラム（特に、非同期的に実行されるメモリー常駐のプログラム）。すなわち、これらのTSRプログラムはフォアグランド・プログラム（システムの制御内のメイン・プログラム）に依存せずにシステムを実行し、管理します。たとえば、フォアグランド適用業務が非活動状態の時に「バックグランドで」タスクを実行する、ネットワーク・ユーティリティーなどがこのタイプのプログラムになります。未処理のネットワーク読み取り要求があるプログラムが保留され、別のプログラムと置き換えられると、ネットワーク・ユーティリティーは要求されたデータを置き換えられた次のプログラムのコンテキストに複写してしまいます。このデータは、この置き換えられたプログラムにより使用されるコードあるいはデータを上書きし、システムの誤動作の原因になることがあります。

- 一部のタイプのメインフレーム通信ソフトウェア。メインフレーム通信ソフトウェアは通常、別のシステムのソフトウェアと通信します。

- システム上で動作している各プロセスに対して個別のデータを維持しているソフトウェア（プロセスとは、１つのセッション内で動作しているすべてのプログラムです）。たとえば、このタイプのソフトウェアには、DOSの機能呼び出しをトラップして、ネットワーク・ドライブへのアクセスを提供するが、DOSにより維持されている内部データは使用しない、ネットワーク・リダイレクターなどがあります。セッション切り替え時に、タスク・スワッパーがこれらのデータに変更を加えている間、これらのテーブルに依存しないリダイレクターはそれ自体でこれらの情報を維持できなければなりません。したがって、このリダイレクターは、セッション切り替えが起きる際には、そのことを通知されなければなりません。また、それ

がセッション切り替えを適切に処理できるまでは、この切り替えを防止したり遅らせたりできなければなりません。

DOSのタスク・スワッピング・プロトコルは、このようなタイプのソフトウェアがこのプロトコルに従うタスク・スワッパーと共存できるようにします。プロトコルは、タスク・スワッパーとシステム上のそのほかのソフトウェアとの間のAPIを指定し、セッション切り替えによって悪影響を受けるソフトウェアが切り替えのタイミングを制御したりそれを防止したりすることができるようにします。そして最終的に、このプロトコルは、タスク・スワッパーがシステム上に同時に存在する場合に発生する問題を最小化して、タスク・スワッパーと相互に協力するための標準方法を提供します。本付録で解説するタスク・スワッピング・プロトコルに従うプログラムは、DOS/VのDOSシェルで用意されているタスク・スワッパーと安全に共存できます。

本付録では、タスク・スワッピング環境での安全な作業のために必要な要件について説明します。

DOSのタスク・スワッピング・プロトコルがタスク・スワッパーとそのほかのプログラム用に標準方法を指定するので、それらは相互に協力できます。タスク・スワッパーとそのほかのタイプのプログラムは、タスク・スワッパーがサーバー、他のプログラムがクライアントとして働く、クライアント／サーバー関係で存在します。本章では、クライアントという用語はこのプロトコルに従い、タスク・スワッパーではないプログラムを指します。上述したように、ほとんどのタスク・スワッパーは、あるセッションを保留し、そのセッションをディスクに移動し、同じアドレス空間に別の適用業務をロードすることによって働きます。ただし、ほとんどのシステムには、DOS、デバイス・ドライバー、およびスワップされないTSRユーティリティーによって占有されるメモリーなど、スワップ対象になるメモリーとは別のエリアがあります。たいていの場合、このメモリーを占有するプログラムは、タスク・スワッパーの前にロードあるいは実行され、どの適用業務をタスク・スワッパーがシステム・メモリーにロードしたかにかかわらず、動作し、アクセス可能です。これらのタイプのプログラムを、グローバル・ソフトウェアといいます。また、タスク切り替えを通じて変更されないメモリーをグローバル・メモリーといいます。一方、ローカル・メモリーとは、タスク・スワッパーにより生成されるセッションに関連するメモリーのことです。タスク・スワッパーがセッションをスワップすると、ローカル・メモリーはディスクにスワップされます。

クライアントの初期化

タスク・スワッパーの最初の役割の１つは、割り込みベクトルの取得INT 21H関数（AH=35H）と割り込みベクトルの設定INT 21H機能（AH=25H）を呼び出すことによって、タスク・スワッパー自体をINT 2FHハンドラーのチェーンに追加することです。クライアント・プログラムは、自身が割り込みチェーンに導入された後、Detect Swapper INT 2FHコールイン関数を呼び出して、タスク・スワッパーがすでに存在しているかどうかを判別する必要があります。タスク・スワッパーが存在している場合、この関数はES:DIにタスク・スワッパーのコールイン・アドレスをセットして戻ります。このES:DIのコールイン・アドレスを使用して、クライアントはHook Call-outコールイン関数を呼び出します。これによりタスク・スワッパーがクライアントのコールアウト関数ハンドラー・アドレスを、クライアントが動作しているセッション用に維持しているコールアウト・ハンドラーのチェーンに追加することができます。

クライアントINT 2FHハンドラー

タスク・スワッパーはINT 2FHを出して、Build Call-out Chain関数とIdentity Instance Data関数という２つの関数を呼び出します。これらの関数は２つとも、タスク・スワッパーによって呼び出され、データ構造体のリンク・リストを作成します。関数に応じて、各構造体は特定のクライアントのコールアウト関数ハンドラー・アドレスか、クライアントのインスタンス・データのどちらかを意味します。一般に、クライアントINT 2FHハンドラーは、以下の手続きにより、自身の構造体をリストに追加します。

1. INT 2FHハンドラーは、呼び出しがタスク・スワッパー・コールアウト関数であるかをAXレジスターの値により判別します。タスク・スワッパー・コールアウト関数でない場合は、farジャンプを使用して制御を以前のINT 2FHハンドラーに転送します。
2. AXによりタスク・スワッパー・コールアウト関数であると識別すると、INT 2FHハンドラーは、フラグをプッシュして(pushf)、以前のINT 2FHハンドラーへfar呼び出しを行います。
3. far呼び出しから戻ったら、INT 2FHハンドラーはクライアントのデータ構造体の適切なフィールドにES:BX（自分の直前のハンドラーのデータ構造体へのポインター）の値を保存します。
4. INT 2FHハンドラーは、クライアント自身のデータ構造へのポインターをES:BXに入れ、割り込みから戻ります(iret)。INT 2FHと使用されているデータ構造体により、特定のINT 2FHコールアウト関数に応じて特定の処置が取られます。

セッション切り替えの保留状態への応答

クライアントは、セッション切り替えがそのクライアントあるいは他のソフトウェアの状態が不安定なときに行われることのないようにすることができなければなりません。このような場合にセッション切り替えが起こると、データの損失や損傷を招くことがあるからです。

タスク・スワッパーはセッション切り替えが実行される前に、２つのコールアウト関数を呼び出します。Query Suspend関数は、タスク・スワッパーが現在活動状態のセッションを保留する準備をしていることを、影響を受けるクライアントに通知します。この関数の呼び出しを受け取ると、クライアントは、必要な作業を実行してセッションの切り替えを進めることもできるし、セッション切り替えを防止することもできます。

タスク・スワッパーは、Query Suspend呼び出しに失敗するクライアントがなければ、次に各クライアントに対してSuspend Session関数を呼び出します。割り込みはタスク・スワッパーがQuery Suspend呼び出しを行っている間はイネーブルであるため、システム状態はクライアントが呼び出しを受け取った後で変更されることがあります。Suspend Session呼び出しにより、クライアントにセッションの切り替えを防止する最後の機会が与えられます。タスク・スワッパーはこの関数を割り込みがディセーブルの状態で呼び出すため、システム状態は、セッション切り替えが実行されるまでは、後に続く他のクライアントによって変更されてはなりません。つまり、割り込みはすべてのクライアントへの呼び出しから戻るまではディセーブルのままでなければなりません。

また、クライアントはソフトウェア割り込みを発行したり、割り込みをイネーブルにする呼び出しを使用したりすることはできません。クライアントはAXにゼロを戻して、セッション切り替えを許可するか、または、１を戻してセッション切り替えを防止するかのどちらかを選択できます。他のレジスターはすべて、保持されている必要があります。

非同期APIを処理するクライアントはQuery SuspendまたはSuspend Session呼び出しを失敗させるときには、自分以外にその適用業務プログラム・インターフェース（API）の要求をより満足する他のクライアントがいないことを、まず確認しなければなりません。クライアントはこれをQuery API Supportコールイン関数を呼び出すことにより判別します。

Query SuspendあるいはSuspend Session呼び出しに失敗するクライアントがある場合でも、タスク・スワッパーによって呼び出されるすべてのクライアントは、保留される予定だったセッションのSession Active呼び出しを受け取ることがあります。このため、保留状態でない、あるいは（保留後）活動化されていないセッションのSession Active呼び出しを受け取ることがあり得ます。クライアントはこれらの呼び出しを無視してかまいません。

Suspend Session呼び出しに失敗するクライアントがない場合、タスク・スワッパーは、割り込みをイネーブルにする前に、現行の割り込みベクトル・テーブルを保管されたコピーに置き換えます。保管されたコピーとは、タスク・スワッパーが最初に起動したときのグローバル状態を表します。これにより、保留されているセッションのローカルな割り込みハンドラーは、そのセッションが再開されるまで呼び出されないことが保証されます。

Suspend Session呼び出しと次のActivate Session呼び出しまでの間、割り込みは断続的にイネーブルにされ、グローバル・ソフトウェアはこの時に割り込みを受け取ることができます。グローバル・クライアントは、Suspend SessionとActivate Session呼び出しまでの間でグローバルでないメモリーの内容を仮定することはできません。

タスク・スワッパーは影響されるクライアントにActivate Sessionコールアウト関数を呼び出して、タスクがまもなく再開されることを知らせます。そして、以前に保留されたセッション（そのローカル・メモリーと割り込みベクトル・テーブルを含む）がメモリーにロードされ、割り込みがイネーブルされたときにSession Active関数を呼び出します。

新しいセッションの作成の保留状態への応答

クライアントは、セッションの切り替えを防止できるのと同じように、タスク・スワッパーが新しいセッションを作成しないようにすることもできます。新しいセッションを作成する前に、タスク・スワッパーはCreate Sessionコールアウト関数を呼び出します。クライアントは新しいセッションに対しての準備をするため適切な処置（たとえば、そのセッションの情報を保持するための追加メモリーの割当てなど）を取ってから、タスク・スワッパーが新しいセッションを作成できるようにします。あるいは、クライアントはCreate Sessionを失敗して、新しいセッションが作成されないようにすることもできます。たとえば、セッション・データを維持している固定長バッファーがい

っぱいで、他のセッション用に使用できないような場合に、クライアントは、新しいセッションが作成されないようにします。

Create Session呼び出しに失敗するクライアントがあると、タスク・スワッパーは一部あるいはすべてのクライアントに対してDestroy Session関数を呼び出します。Destroy Session関数はクライアントにCreate Session呼び出しで渡したセッションIDが無効になったことを通知します。タスク・スワッパーはすべてのクライアント、あるいはCreate Session呼び出しを受け取ったクライアントに対して呼び出しを行います。このため、まだ作成あるいは起動されていないセッションのDestroy Sessionを受け取る可能性があります。クライアントはこのDestroy Session呼び出しを無視してかまいません。

Create Session呼び出しを失敗するクライアントがない場合、タスク・スワッパーは通常現行セッションを保留してから、クライアントに新しいセッションが管理されることを通知するために、あるフラグをセットしてActivate Session機能を呼び出します。これにより、クライアントは新しいセッションに対する準備をするために必要な何らかの処置を取る機会が与えられます。そして、タスク・スワッパーはSession Activeを呼び出し（再度フラグ・セットを使用する）、クライアントに新しいセッションが起動されたことを通知します。ただし、セッション・マネージャーの中には、ユーザーが非活動状態でセッションを起動できるようにするものもあるため（DOSシェルやWindows[*1]など）、Activate Session呼び出しおよびSession Active呼び出しは、Create Session呼び出しのすぐ後で起こるとは限りません。新しく作成されたセッションが最初に活動状態になる前に他のセッションが起動され、保留されることもあり得ます。

クライアントの終了

終了する前に、クライアントは次の２つの手続きを実行する必要があります。

- Unhook Call-Outというコールイン関数を呼び出さなければなりません。これにより、タスク・スワッパーにより維持されているローカル・コールアウト・ハンドラー・チェーンからクライアントのコールアウト関数ハンドラーが削除されます。
- クライアントは、割り込みベクトルの設定INT 21H機能（AH=25H）を呼び出すことによって、INT 2FHハンドラーのチェーンからクライアントのINT 2FHハンドラーを削除し、以前のINT 2FHハンドラー（クライアントが初期化されるときに保管されたもの）のアドレスにINT 2FH割り込みベクトルを置き換えなければなりません。

[*1] WindowsはMicrosoft Corporationの製品です。

Switch_Call_Back_Infoデータ構造体

すべてのクライアントはSwitch_Call_Back_ Info（SCBI）データ構造体を維持していなければなりません。クライアントは、いくつかの異なる関数を呼び出したりそれらに応答するときに、このデータ構造体をタスク・スワッパーに渡します。SCBI構造体には、クライアントのコールアウト関数ハンドラーのエントリー・ポイントと、API_Info_Strucデータ構造体へのポインターに関する情報が格納されています。これらの構造体により、クライアントがサポートする非同期APIのタイプと提供できるサポートのレベルが指定されます。Switch_Call_Back_Infoデータ構造体は、以下のように定義されます。

```
Switch_Call_Back_Info STRUC

SCBI_Next             dd  ?
SCBI_Entry_Pt         dd  ?
SCBI_Reserved         dd  ?
SCBI_API_Ptr          dd  ?

Switch_Call_Back_Info ENDS
```

Switch_Call_Back_Info構造体には、以下のフィールドがあります。

- SCBI_Next – この32ビット値には、クライアント・チェーンの中の次の構造体を指すポインターがはいっています。タスク・スワッパーはBuild Call-Out Chain INT 2FHコールアウト関数を呼び出し、このチェーンを構築します。
- SCBI_Entry_Pt – この32ビット値には、クライアントのコールアウト関数ハンドラーのエントリー・ポイントを指すポインターがはいっています。
- SCBI_Reserved – この32ビット値は、タスク・スワッパーが使用するために予約済みです。
- SCBI_API_Ptr – この32ビット値には、API_Info_Strucデータ構造体のゼロで終了するリストを指すセグメント:オフセット・ポインターが含まれます。これらの構造体により、クライアントがさまざまな非同期API用に提供するサポートのタイプが指定されます。

API_Info_Strucデータ構造体

API_Info_Struc（AIS）データ構造体には、クライアントが特定のタイプの非同期APIに提供するサポートのレベルに関する情報が格納されます。API_Info_Struc構造体は次のように定義されます。

```
API_Info_Struc STRUC

AIS_Length         dw 10
AIS_API            dw ?
AIS_Major_Ver      dw ?
AIS_Minor_Ver      dw ?
AIS_Support_Level  dw ?

API_Info_Struc ENDS
```

API_Info_Strucデータ構造体には、以下のフィールドがあります。

1. AIS_Length – この16ビット値は、AISデータ構造体の長さを指定します。
2. AIS_API – この16ビット値は、クライアントによってサポートされる非同期APIのIDを指定します。以下の値が定義されます。

```
API_NETBIOS (1h)        NETBIOS

API_8022 (2h)           802.2

API_TCPIP (3h)          TCP/IP

API_LANMAN (4h)         LAN Manager named pipes

API_IPX (5h)            NetWare IPX
```

- AIS_Major_Ver – この16ビット値は、AIS_Support_Levelフィールドにより指定されたレベルのサポートを提供するクライアントのAPIの最高バージョンのメジャー部分を指定します。たとえば、クライアントが指定されたレベルでサポートするAPIの最高バージョンが3.10の場合、このフィールドは3hに設定されます。
- AIS_Minor_Ver – この16ビット値は、指定されたレベルのサポートを提供するクライアントのAPIの最高バージョンのマイナー部分を指定します。たとえば、指定されたレベルでクライアントがサポートするAPIの最高バージョンが3.10の場合、このフィールドは、Ahに設定されます。
- AIS_Support_Level –この16ビット値は、クライアントがAPIの特定のバージョン用に提供するサポートのレベルを指定します。このフィールドの値の範囲と意味は、特定のAPIによって決まります。NETBIOSには、以下の定義が使用されます。
  - 最小限のサポート。クライアントは、適用業務が一度でも非同期APIを実行した場合、たとえ要求が完了した後であっても、セッション切り替えを防止します。
  - APIレベル・サポート。クライアントは未処理の非同期要求を追跡し、その時にタスク切り替えが行われるのを防ぎます。クライアントは、未処理の非同期要求がすべて完了すると、タスク切り替えを許可します。
  - スワッパー互換性。API提供元は、非同期要求が未処理の場合であっても、切り替えを許可します。ただし、これは、バッファー・サイズなどにより制限され、失敗する要求もあります。
  - シームレス互換性。API提供元は、常にセッション切り替えを許可します。これによりデータの損失が発生することはありません。

Win386_Startup_Info_Strucデータ構造体

Win386_Startup_Info_Strucデータ構造体は次のように定義されます。

```
Win386_Startup_Info_Struc      STRUC

SIS_Version                        db  3,0 ; ignored
SIS_Next_Dev_Ptr                   dd  ?   ; Ptr to previous handler's
                                           ; Win386_Startup_Info_Struc
SIS_Virt_Dev_File_Ptr              dd  0   ; ignored
SIS_Reference_Data                 dd  ?   ; ignored
SIS_Instance_Data_Ptr              dd  ?   ; Ptr to IIS structures

Win386_Startup_Info_Struc ENDS
```

Win386_Startup_Info_Strucは、Microsoft Windows起動INT 2FH関数に応答するために使用される構造体と同じです。ただし、DOSのタスク・スワッパーはSIS_Next_Dev_File_PtrフィールドおよびSIS_Instance_Data_Ptrフィールドのみを使用します。

Win386_Startup_Info_Strucデータ構造体には、以下のフィールドがあります。

- SIS_Version– この２バイトのフィールドは使用しません。
- SIS_Next_Dev_Ptr –この32ビット値には、クライアント・チェーンの次の構造体を指すセグメント:オフセット・ポインターが含まれます。このチェーンの構成方法の詳細については、以降のページを参照してください。
- SIS_Virt_Dev_File_Ptr – この32ビットのフィールドは使用されません。
- SIS_Reference_Data – この32ビットのフィールドは使用されません。
- SIS_Instance_Data_Ptr – この32ビット値には、Instance_Item_Strucデータ構造体のリストを指すセグメント:オフセット・ポインターが含まれます。各構造体はインスタンス・データの１つの連続ブロックを記述します。このリストは32ビットのゼロ値で終了します。

Instance_Item_Strucデータ構造体

Instance_Item_Strucデータ構造体は次のように定義されます。

```
Instance_Item_Struc STRUC

IIS_Ptr           dd  ?
IIS_Size          dw  ?

Instance_Item_Struc ENDS
```

Instance_Item_Struc構造体には、以下のフィールドがあります。

- IIS_Ptr – この32ビット値には、インスタンス・データのブロックの最初のバイトを指すセグメント:オフセット・ポインターが含まれます。
- IIS_Size – この16ビット値は、インスタンス・データのサイズをバイト単位で指定します。

Swapper_Ver_Struc構造体

以下にSwapper_Ver_Strucデータ構造体の定義を示します。

```
Swapper_Ver_Struc   STRUC

SVS_API_Major            dw  ?
SVS_API_Minor            dw  ?
SVS_Product_Major        dw  ?
SVS_Product_Minor        dw  ?
SVS_swapper_ID           dw  ?
SVS_Flags                dw  ?
SVS_Name_Ptr             dd  ?
SVS_Prev_Swapper         dd  ?

Swapper_Ver_Struc   ENDS
```

Swapper_Ver_Struc構造体には、以下のフィールドがあります。

- SVS_API_Major – この16ビット値はスワッパーがサポートするタスク・スワッピング・プロトコルの最高バージョンのメジャー部分を指定します。たとえば、クライアントがサポートするプロトコルの最高バージョンが3.10の場合、このフィールドは、3hに設定されます。現行バージョンは1.0です。
- SVS_API_Minor – この16ビット値はスワッパーがサポートするタスク・スワッピング・プロトコルの最高バージョンのマイナー部分を指定します。たとえば、タスク・スワッパーがサポートするプロトコルの最高バージョンが3.10の場合、このフィールドはAhに設定されます。現行バージョンは1.0です。
- SVS_Product_Major – この16ビット値はSVS_API_Majorと同じ形式のタスク・スワッパーのメジャー・バージョンを指定します。
- SVS_Product_Minor – この16ビット値はSVS_API_Minorと同じ形式のタスク・スワッパーのマイナー・バージョンを指定します。
- SVS_Swapper_ID – この16ビットのフィールドは、その下位４ビットで、Allocates関数から得たスワッパーID値を指定します。
- SVS_Flags– この16ビットのフィールドには、フラグとして使用される16ビット配列が含まれます。タスク・スワッピング・プロトコルのこのバージョンでは、ビット０だけが使用されます。スワッパーが現在使用不可能である場合は、ビット０が１にセットされます。使用可能である場合は、ビット０はゼロです。
- SVS_Name_Ptr – この32ビット値には、タスク・スワッパーを識別するゼロで終了するASCII文字列を指すセグメント:オフセット・ポインターが入ります。
- SVS_Prev_Swapper –この32ビット値には、スワッパー検出INT 2FHタスク・スワッパー関数により戻された、以前にロードされたスワッパーのコールアウト関数のエントリー・ポイントのアドレス（セグメント:オフセット形式）が入ります。

関数の説明

本節では、DOSのタスク・スワッピング・プロトコルを構成する、INT 2FHハンドラー関数、タスク・スワッパー・コールアウト関数、およびタスク・スワッパー・コールイン関数について説明します。関数説明は、これらのカテゴリーにしたがってグループ分けして行います。各カテゴリーでは、関数は番号順で説明します。各関数説明は、関数名、関数の簡単な説明、関数の開始および終了の必要条件という順序で行います。その他の解説は、入力および終了情報の後に記載されています。

注: これらの関数によりパラメーターとして使用されないすべてのレジスターは、保持されていなければなりません。

タスク・スワッパーINT 2FHハンドラー関数

現在、タスク・スワッパーには、以下に示されるような１つのINT 2FH関数があります。

Detect Swapper（スワッパー検出　AX = 4B02H）

クライアントは、スワッパー検出INT 2FH関数を呼び出して、タスク・スワッパーが現在動作中であるかどうかを判別し、そのコールイン（タスク・スワッパーへの）関数エントリー・ポイントのアドレスを得ます。

例:

```
        MOV  AX,04B02H          ;Detect presence of a task-swapper
        XOR  BX,BX              ;Required for future extensibility
        XOR  DI,DI
        MOV  ES,DI
        INT  2FH
             ;
             ;Save call-out address to the current task-swapper
             ;
        MOV  WORD PTR OUT_TO_SWAPPER+0, DI
        MOV  WORD PTR OUT_TO_SWAPPER+2, ES

--------

OUT_TO_SWAPPER DD   0    ; Call-out address to the current
task-swapper

All other registers are preserved.
```

解説

ES:DIに非NULLポインターが戻された場合は、タスク・スワッパーが存在することを示します。AXは将来の拡張可能性のためにゼロで戻され、キャリー・フラグはクリアになります。

クライアントは、自身の初期化中にこの関数を呼び出し、スワッパーが検出された場合は、適切な処置を取ります。

クライアントINT 2FHハンドラー関数

このプロトコルに完全に準拠しているクライアントは、タスク・スワッパーからBuild Call-out Chain関数（4B01H）に正しく応答できるINT 2FHハンドラーを持たなければなりません。さらに、クライアントが各セッション用のデータを維持している場合は、Identify Instance Data関数（4B05H）にも応答する必要があります。以下の節では、これらの関数を説明します。

Build Call-out Chain（コールアウト・チェーン構築）

このクライアントINT 2FHハンドラー関数により、クライアントのコールアウト関数へのエントリー・ポイントをスワッパー上のテーブルに登録します。

```
On Entry:
        ; Respond to Call_Out function from task-swapper

        cmp     ax,04b01h               ;Build Chain Call-in function?
        jnz     Pass_Function_On
        test    Bit_flag,00000001b
        jnz     Pass_Function_On
          ;
          ;ES:BX = 0:0 for future extensibility
          ;CX:DX = call-in address of the calling task-swapper
          ;
        pushf
        call    far Prev_Int2f_Handler

          ;
        mov     offset SCBI_Next,bx
        mov     offset SCBI_Next+2,es

Back_to_swapper:

        and     Bit_flag,1           ;We've been here
        mov     bx,offset Swap_Call_Back_Info
        mov     es,cs
          ;
          ;All other registers must be preserved.
          ;
        iret
```

```
Pass_Function_On:
          ;
        jmp     far Prev_Int2f_Handler

--------

Bit_flag        db      0               ; bit 1 = 0 if not called by swapper
                                        ;       = 1 if called by the swapper
```

解説

メモリーにロードされる最初のクライアントである場合は、ES:BX＝0:0となります。

タスク・スワッパーはこの関数を呼び出して、すべてのグローバル・クライアント（現行セッションで動作しているクライアント）のコールアウト関数エントリー・ポイントと、各クライアントにサポートされている非同期APIに関する情報のリンク・リストを作成します。この関数が戻るとき、ES:BXには、この情報を含むクライアントのSCBIデータ構造体を指すポインターがはいります。

```
Swap_Call_Back_Info STRUC

        SCBI_Next          dd  ?       ;pointer to next structure in list
        SCBI_Entry_Pt      dd  ?       ;CS:IP of entry point procedure
        SCBI_Reserved      dd  ?       ;used by the swapper
        SCBI_API_Ptr       dd  ?       ;pointer to list of API structures

Swap_Call_Back_Info ENDS
```

SCBIデータ構造体の説明に関しては、本書のE-6ページの『Switch_Call_Back_Infoデータ構造体』を参照してください。

クライアントは、INT 2FHを受け取ると、AXをチェックして、そのINT 2FHがクライアントINT 2FHハンドラー関数であるかどうかを判別します。そうでない場合は、クライアントはfarジャンプを使用して制御を以前のINT 2FHに渡します。クライアントINT 2Fhハンドラー関数である場合は、クライアントは即座にフラグをプッシュし、far呼び出しを使用して以前のINT 2FHベクトルを呼び出します。クライアントは呼び出しを行うまでは、いずれのレジスターも変更してはいけません。

この呼び出しから戻るとき、ES:BXには、直前のクライアントのSCBIデータ構造体のアドレス（あるいは現行クライアントがメモリーに最初にロードされた場合は０:０）がはいります。ES:BXが０:０であるかどうかにかかわらず、ES:BX内の値をクライアントのSCBIデータ構造体のSCBI_Nextフィールドに保存します。そしてクライアントは、自身のSCBIデータ構造体のアドレスをES:BXに入れ、割り込みから戻ります。

この呼び出しが呼び出したタスク・スワッパーに戻るとき、ES:BXは、メモリーに最後にロードされたクライアントのSCBIデータ構造体を指します。その結果、最も新しくロードされたクライアントのSCBIデータ構造体がチェーンの先頭に現れます。したがって、最も新しくロードされたクライアントは、それ以前にロードされたクライアントよりも前に呼び出されます。たとえば、これによりセッション・スワップが発生した場合、未処理の非同期要求のある適用業務はネットワーク・プログラム（これも１つのクライアント）に対してコールアウト関数が呼び出される前に、その要求を事前に取り消すことができます。呼び出しが逆の順序で行われた場合、未処理の非同期要求のために、ネットワーク・プログラムがセッション・スワップを防止してしまうかもしれません。

Build Call-out Chain関数を呼び出す際、CX:DXには、呼び出したタスク・スワッパーのコールイン関数エントリー・ポイントがはいります。クライアントは、INT 2FHを処理する間に、実行する関数に相当する引き数でこのルーチンを呼び出すことができます。ただし、このINT 2FH呼び出しで渡されるアドレスは、後のコールアウト関数呼び出しで使用できるようになるアドレスと同じであるとは限りません。

Identify Instance Data（インスタンス・データ識別）

このクライアントINT 2FHハンドラー関数は、クライアントによって保持されるインスタンス・データを識別します。以下に例を示します。

```
On Entry:
        ; Respond to Call_Out function from task-swapper for Instance Data

       cmp    ax,04b05h               ;Build Chain Call-in function?
       jnz    Pass_Function_On
         ;
         ;ES:BX = 0:0 for future extensibility
         ;CX:DX = call-in address of the calling task-swapper
         ;Calls to DOS can be made

          ;Make call to previous client's Win386_Startup_Info_Struc
          ; data structure
          ;
        pushf
        call  far Prev_Int2f_Handler

        mov   offset SIS_Next_Dev_Ptr,BX
        mov   offset SIS_Next_Dev_Ptr+2,ES

Back_to_swapper:
          ;
          ;Provide Win386_Startup_Info_Struc data structure address
          ;
        mov   bx,offset Win386_Startup_Info_Struc
        push  cs
        pop
        iret                         ;All other registers must be preserved

Pass_Function_On:
        jmp   far Prev_Int2f_Handler ;to previous INT 2fh handler

--------


Win386_Startup_Info_Struc      STRUC

SIS_Version                 db  3,0 ; ignored
SIS_Next_Dev_Ptr            dd  ?   ; Ptr to previous handler's
                                    ;   Win386_Startup_Info_Struc
SIS_Virt_Dev_File_Ptr       dd  0   ; ignored
SIS_Reference_Data          dd  ?   ; ignored
SIS_Instance_Data_Ptr       dd  ?   ; Ptr to IIS structures

Win386_Startup_Info_Struc ENDS
```

解説

タスク・スワッパーはこの関数を呼び出して、システムで動作しているすべてのクライアントのインスタンス・データ・ブロックのリンク・リストを作成します。

クライアントはINT 2FHを受け取ると、INT 2FHはAXをチェックして、クライアントINT 2FHハンドラー関数を呼び出しているかどうか調べます。呼び出していない場合は、クライアントはfarジャンプを使用して、以前のINT 2FHハンドラーに制御を渡します。呼び出している場合は、クライアントINT 2FHハンドラー・ルーチンは即座にフラグをプッシュし、far呼び出しを使用して以前のINT 2FHベクトルを呼び出します。クライアントは呼び出しを行うまでレジスターを変更してはいけません。

呼び出しから戻るとき、ES:BXには、直前のクライアントのWin386_Startup_Info_Strucデータ構造体のアドレス（あるいは、現在のクライアントがメモリーに最初にロードされた場合は、０:０）が収められています。ES:BXが０:０であるかどうかにかかわらず、ES:BXの値をクライアントのWin386_Startup_Info_Strucデータ構造体のSIS_Next_Dev_Ptrフィールドに保存します。そして、クライアントは自身のWin386_Startup_Info_Strucデータ構造体をES:BXに入れ、割り込みから戻ります。

呼び出しが呼び出したタスク・スワッパーに戻るとき、ES:BXはメモリーに最後にロードされたクライアントのWin386_Startup_Info_Strucデータ構造体を指します。

タスク・スワッパー・コールイン関数

DOSのタスク・スワッパーはこのプロトコルに完全に準拠し、以下の５つの関数に正しく応答できるコールイン関数エントリー・ポイントを含んでいます。

- Get Version（バージョン取得　AX = 0H）
- Test Memory Region（メモリー領域テスト　AX = 1H）
- Hook Call-out（コールアウトのフック　AX = 4H）
- Unhook Call-out（コールアウトのフック解除　AX = 5H）
- Query API Support（APIサポートの問合せ　AX = 6H）

以下のセクションでは、これらの関数について説明します。

注:　すべてのタスク・スワッパー・コールイン関数はキャリー・フラグがクリアの状態で戻ります。コールイン関数がキャリー・フラグがセットされた状態で戻った場合は、その関数はタスク・スワッパー内部で正しく実行されていません。

Get Version（バージョン取得）

このタスク・スワッパー・コールイン関数は現在のタスク・スワッパー、そのバージョン番号、およびそれがサポートしているタスク・スワッピング・プロトコルのレベルを示します。

例:

```
        MOV     AX,0            ;Get Version call to the task-swapper
        CLI                     ;Interrupts disabled
        CALL    OUT_TO_SWAPPER  ;From Detect swapper INT 2fh, AX=4b02h
                                ;
                                ;AX = 0 for future extensibility
                                ;All other registers are preserved.
                                ;
                                ;Save address of the swapper's Version
                                ;  Data structure
                                ;
        MOV     SWAPPER_VER_STRUC_PTR+0,BX
        MOV     SWAPPER_VER_STRUC_PTR+2,ES

--------

SWAPPER_VER_STRUC_PTR  DD      0
```

解説

以下にSwapper_Ver_Strucデータ構造体の定義を示します。

```
Swapper_Ver_Struc   STRUC

SVS_API_Major           dw  ?
SVS_API_Minor           dw  ?
SVS_Product_Major       dw  ?
SVS_Product_Minor       dw  ?
SVS_Swapper_ID          dw  ?
SVS_Flags               dw  ?
SVS_Name_Ptr            dd  ?
SVS_Prev_Swapper        dd  ?

Swapper_Ver_Struc   ENDS
```

Swapper_Ver_Strucデータ構造体の詳細については、本書のE-11ページの『Swapper_Ver_Struc構造体』を参照してください。

メモリー領域テスト (Test Memory Region)

このタスク・スワッパー・コールイン関数を使用して、現行セッションに対するグローバル、あるいはローカルのメモリー位置を判別できます。グローバルなメモリーは、タスク・スワップが起こっても置き換えられません。

例:

```
MOV     AX,1                ;Test memory region call-out to the
                            ;  task-swapper
MOV     DI,OFFSET BUFFER    ;Address of buffer to be tested
MOV     ES,SEG BUFFER       ;ES is the buffer's segment address
MOV     CX,BUFFER_LENGTH    ;Length of buffer in bytes (0 to 65535)
                            ;  where 0 indicates 64K bytes (65536)
CLI                         ;Interrupts disabled
CALL    OUT_TO_SWAPPER      ;Obtained from Detect Swapper
                            ;  INT 2fh, AX=4b02h
                            ;
                            ;Carry flag is clear
MOV     REGION_LOCATION,AX
                            ;AX = 0, Buffer is in global memory
                            ;AX = 1, Buffer is partially in global
                            ;        and partially in local memory
                            ;AX = 2, Buffer is in local memory
                            ;
                            ;All other bits are reserved and
                            ; must be 0
                            ;All other registers are preserved.
```

解説

テストされるバッファーが64キロバイトよりも長い場合は、領域すべてをテストするのに複数の呼び出しが必要です。メモリーがグローバルであるか、あるいはローカルであるかは、現在のタスク・スワッパーが判定します。クライアントは以下のそれぞれのQuery Suspend呼び出し、またはSession Active呼び出し内で、この関数によりメモリーがセッション・スワップを行うタスク・スワッパーに対してグローバルであるか、あるいはローカルであるかを判定します。

グローバル・ソフトウェアはこの関数を使用して、グローバル・ソフトウェアの別のエリアから呼び出された非同期呼び出しを識別できます。呼び出し側の適用業務がグローバル・メモリーに属している場合は、グローバル・ソフトウェアはセッション・スワップが起きるときに特別な処置を取る必要はありません。これは、呼び出し側の適用業務のバッファーやコールバック・アドレスが、どのセッションが現在実行中であってもアクセス可能であるためです。

メモリー常駐ユーティリティーはこの関数を使用して、自分自身がセッション内でローカルに実行されているかどうかを判定できます。たとえば通信適用業務は、自分自身が

ローカルであれば、保留状態になる前に一時的にシャットダウンすることができます。ただし、適用業務がグローバルに動作している場合は、セッション・スワップによる影響は受けないため、この処置は必要ありません。

Hook Call-out（コールアウトのフック）

このコールイン関数は呼び出し側のクライアントのコールアウト関数ハンドラーのアドレスをタスク・スワッパーのコールアウト・チェーンに追加します。

次に例を示します。

```
MOV     AX,4                    ;Call-out to swapper to add this
                                ;  client's Call-out address

MOV     DI,OFFSET SWAP_CALL_BACK_INFO
                                ;
MOV     ES,CS                   ;Swap_Call_Back_Info (SCBI)
                                ;  data structure
                                ;Interrupts are enabled.
                                ;Calls to DOS can be made.

CALL    OUT_TO_SWAPPER          ;Obtained from Detect swapper
                                ;  INT 2fh, AX=4b02h
                                ;
                                ;Carry flag is clear.
                                ;
                                ;AX = 0 for future extensibility
                                ;
                                ;All other registers are preserved
```

注: クライアントがこの構造体のSCBI_Nextフィールドに入力することは求められていません。詳細については、本書のE-6ページの『Switch_Call_Back_Infoデータ構造体』を参照してください。

解説

初期化中に、クライアントはスワッパーのDetect Swapper INT 2FH関数を呼び出します。この呼び出しがタスク・スワッパーが動作中であることを示している場合は、このクライアントはこのタスク・スワッパーのHook Call-outコールイン関数を呼び出して、そのコールアウト・ハンドラーをタスク・スワッパーのコールアウト・チェーンに追加します。タスク・スワッパーの中には、各タスク・スワッパー・イベントの前に、そのつどBuild Call-out Chain INT 2FH関数を呼び出すことにより、コールアウト・チェーンを作成するものもありますが、通常のタスク・スワッパーは、このリストを初期化時にのみ作成します。これらのタスク・スワッパーには、各セッション用に個別のチェーンが保持されています。タスク・スワッパーは新しいセッションを作成するたびに、タスク・スワッパーの初期化時に生成されたグローバル・チェーンのコピーを、セッションに与えます。（タスク・スワッパーは１つのグローバル・チェーンと各セッション毎の個別のローカル・チェーンを保持できます。）セッションが作成されると、そ

のセッション内でローカルに動作しているクライアントは、コールアウトのHook Call-in関数を呼び出すことによって、そのコールアウト・ハンドラーのアドレスをローカル・チェーンに明示的に追加する必要があります。

クライアントは、終了する前にそれ自体をコールアウト・チェーンから明示的にフック解除しなければなりません。Unhook Call-outコールイン関数は、クライアントをタスク・スワッパーのコールアウト・チェーンからフック解除します。

コールアウトのフック解除 (Unhook Call-out)

このコールイン関数は呼び出し側のクライアントのコールアウト関数ハンドラーのアドレスをタスク・スワッパーのコールアウト・チェーンから削除します。

例:

```
MOV     AX,5                    ;Call-in to swapper to remove
                                ;  this client's call-out address
                                ;
MOV     DI,OFFSET SWAP_CALL_BACK_INFO
MOV     ES,CS                   ;Swap_Call_Back_Info (SCBI)
                                ;  data structure
                                ;Interrupts are enabled.
                                ;Calls to DOS can be made.
CALL    OUT_TO_SWAPPER          ;Obtained from Detect swapper
                                ;INT 2fh, AX=4b02h
                                ;
                                ;Carry flag is clear.
                                ;
                                ;AX = 0 for future extensibility
                                ;
                                ;All other registers are preserved
```

詳細については、本書のE-6ページの『Switch_Call_Back_Infoデータ構造体』を参照してください。

解説

初期化中に、クライアントはスワッパーのDetect Swapper INT 2FH関数を呼び出します。この呼び出しがタスク・スワッパーが実行中であることを示している場合は、このクライアントは、タスク・スワッパーのHook Call-outコールイン関数を呼び出して、そのコールアウト・ハンドラーをタスク・スワッパーのコールアウト・チェーンに追加します。そして、終了する前に、クライアントはUnhook Call-out関数を明示的に呼び出して、それ自体をコールアウト・チェーンから削除しなければなりません。

Query API Support（APIサポート問合せ）

このコールアウト関数は、セッション・スワッピングを制御して特定の非同期APIを処理する必要があるかどうかをクライアントに示します。

例:

```
        MOV     AX,6                    ;Call-out to swapper for most
                                        ;  capable API handler
        MOV     BX,????                 ;ID value of the asynchronous API


        CALL    OUT_TO_SWAPPER          ;Obtained from Detect swapper
                                        ;  INT 2fh, AX=4b02h

        MOV     BEST_API_SUPPORTER+0,BX
        MOV     BEST_API_SUPPORTER+2,ES

                                        ;Carry flag is clear.
                                        ;AX = 0 for future extensibility
                                        ;All other registers are preserved

--------


API_Info_Struc STRUC

AIS_Length        dw    ?               ;length of the structure
AIS_API           dw    ?               ;the API ID value
AIS_Major_Ver     dw    ?               ;major version of API spec
AIS_Minor_Ver     dw    ?               ;minor version of the API spec
AIS_Support_Level dw    ?               ;support level

API_Info_Struc ENDS


Note:  See E-7ページの『API_Info_Strucデータ構造体』 in this manual
       for more information and the description of the AIS_API field.
```

解説

この関数は、どのクライアントが（特定の非同期APIに関して）セッション・スワッピングを制御するのかを判別します。クライアントがQuery SuspendあるいはSuspend Sessionコールアウト関数を処理している場合に、非同期APIの状態が原因でセッション・スワップを防止すべきと判断したとします。その際にはまず、Query API Supportコールイン関数を呼び出して、このクライアントがAPIを処理するのに最も適したものであるかどうかを判定しなければなりません。そのクライアントが最も適したクライアントである場合は、セッション・スワップが防止され得ます。最も適したクライアント

ではない場合は、セッション・スワップを防止すべきでなく、もし必要があれば、代わりにより適したクライアントがセッション・スワップを防止します。

非同期APIにはすべて、ID値が割り当てられています。サポートのレベルは、数値によって定義され、識別されます。割り当てられているサポートのレベルが高いほど、そのクライアントがセッション・スワッピングを許可する状況も多くなります。

クライアント・プログラムには、それらがサポートする非同期APIとそれぞれのサポートのレベルに関する情報が保持されます。これは、API_Info_Strucデータ構造体のリストの中にあります。クライアントは、Swap_Call_Back_Info（SCBI）データ構造体のリストの先頭を指すポインターを提供します。API_Info_Struc構造体の詳細については、本書のE-7ページの『API_Info_Strucデータ構造体』 を参照してください。SCBIデータ構造体の詳細については、本書のE-6ページの『Switch_Call_Back_Infoデータ構造体』 を参照してください。

この関数は最も適したクライアントのAPI_Info_Strucデータ構造体のアドレスを戻すことによって、特定のAPIをサポートするのに最も適したクライアントを示します。最も適したクライアントとは、APIの最も高いバージョンをサポートするクライアントのことです。複数のクライアントが同じ最も高いバージョンをサポートしている場合、最も適したクライアントとは、そのバージョンをサポートするレベルが最も高いクライアントのことです。

この関数によって戻されるAPI_Info _Strucアドレスが、呼び出したクライアント自身のAPI_Info_Strucと同じである場合、このクライアントはセッション・スワップを防止し得ます。

タスク・スワッパー・コールアウト関数

このプロトコルに準拠するクライアントには、以下の８つの関数に正しく応答できるコールアウト関数エントリー・ポイントが含まれています。

- Init swapper（スワッパー初期化　AX = 0H）
- Query Suspend（保留問合せ　AX = 1H）
- Suspend Session（セッション保留　AX = 2H）
- Activate Session（セッション起動　AX = 3H）
- Session Active（セッション活動状態　AX = 4H）
- Create Session（セッション作成　AX = 5H）
- Destroy Session（セッション破棄　AX = 6H）
- Swapper Exit（スワッパー終了　AX = 7H）

クライアントは特別な関数を用意する必要はなく、制御を呼び出し側のタスク・スワッパーに戻すことによって、これらの関数呼び出しのいずれにも応答できます。以下の節ではこれらの関数について説明します。

Init swapper（スワッパー初期化）

このタスク・スワッパー・コールアウト関数は、新しいタスク・スワッパーが初期化されていることをクライアントに通知します。

例:

```
        CMP     AX,0        ;Init swapper call?
                            ;ES:DI = the call-out address to the
                            ;  calling task-swapper.
                            ;Interrupts are enabled.
                            ;Calls to DOS can be made.

                            ;The task-swapper can safely load, nonzero value
                            ;  indicates that the task-swapper should not load.

        MOV     AX,OKAY_TO_SWAP

        RET                 ;Return to task-swapper

--------

OKAY_TO_SWAP    DB    0    ;Flag to indicate if its okay to swap away
```

解説

この関数を呼び出すのは必ずしもタスク・スワッパーではないため、クライアントはES:DIで渡されるコールアウト・アドレスがその後のコールイン関数で渡されるアドレスと同じであるとはみなしてはいけません。このアドレスはNULLであることもあります。

セッション・マネージャー適用業務、あるいはセッション・スワッピングをサポートする環境は、初期化時にこの関数を呼び出す必要があります。タスク・スワッパーと共存するために特別な処置を取る必要のあるグローバル・クライアントは、この呼び出しを受け取るとその処置を取ります。ES:DIで提供されるコールイン・エントリー・ポイントはGet Versionコールイン関数に応答できなければなりません。

通常は、タスク・スワッパー自体ではなく、タスク・スワッパーを呼び出し制御する適用業務が、Init swapperコールアウト関数を呼び出します。たとえば、DOS/VのDOSシェルはその初期化時に、実際にセッション・スワッピングを実行するDOSタスク・スワッパーが起動する前に、Init swapperコールアウト関数を呼び出します。Init swapper呼び出しを失敗する（すなわち、AXの値をゼロ以外で戻す）クライアントがあると、DOSシェルはタスク切り替えを使用不可にします。他のタスク・スワッピング適用業務では、クライアントがこの関数に失敗すると、終了することもあります。Init swapperコールアウト関数呼び出しに失敗するクライアントがあると、そのクライアントも含めて、すべてのクライアントが、Swapper Exitコールアウト関数の呼び出しを受け取ることがあります。この結果、クライアントは先に対応するInit swapper呼

び出しを受けなくてもSwapper Exit呼び出しを受け取ることがあります。クライアントはこのSwapper Exit呼び出しを無視してもかまいません。

query Suspend（保留の問合せ）

このコールアウト関数は、クライアント・プログラムにタスク・スワッパーがセッション・スワッピング用に準備中であるということを通知します。

例:

```
CMP     AX,1                ;Query Suspend check from the task-swapper
JA      CHECK_NEXT_FUNCTION
CMP     BX,OUR_SESSION_ID   ;BX has the current session ID

  ;Interrupts are enabled
  ;Calls to DOS can be made
  ;ES:DI = The call-out address of the calling task-swapper


MOV     AX,SWAP_FLAG
CMP     AX,0
JE      RETURN_TO_CALLER

  ;Determine that the API is not being handled by another,
  ;  more competent client


MOV     AX,6                ;Call-out to swapper for most capable
                            ;  API handler
MOV     BX,????             ;ID value of the asynchronous API
CALL    OUT_TO_SWAPPER

  ;Does the most capable client have the same address as this client?
```

```
        CMP     OFFSET API_INFO_STRUC,BX

        JNE     OTHER_CLIENT_TO_HANDLE
        MOV     AX,1
        JMP     RETURN_TO_CALLER

OTHER_CLIENT_TO_HANDLE:
        XOR     AX,AX

RETURN_TO_CALLER:
        RET

--------

SWAP_FLAG     DB    0    ;0 if a session swap can be performed safely
                         ;1 if the client cannot safely handle a session
                         ;  swap.  All other values are reserved
                         ;
                         ;All other registers must be preserved.
CHECK_NEXT_FUNCTION:
```

解説

タスク・スワッパーは、セッション・スワップが要求されたときにこの関数を呼び出します。クライアントはそのセッション・スワップを防止することができます。あるいは、クライアントは戻る前にスワップのために必要なすべての操作を実行することもできます。

グローバル・クライアントは現在のセッションIDを使用して、セッション・スワップが発生するときに保留されるセッションを識別できます。また、このクライアントはこのIDを使用して、セッションが保留される際にセッションについての情報を保持し、セッションが再開される際に情報を復元することができます。セッションIDはタスク・スワッパーにより提供される任意の値です。この値は連続している必要はなく、また同じ値は１つのセッションが破棄された後に再使用できます。クライアントは、Test Memory Region関数を呼び出して、メモリー内の特定のコードあるいはデータがセッション・スワップにより影響されるかを判別し、セッション・スワップを許可するかどうかを決定します。たとえば、ネットワーク・リダイレクターは、未処理の要求記述子のチェーンを介して動作し、Test Memory Region関数を使用して、バッファーあるいはコールバック・アドレスがローカル・メモリーにあるかどうかをチェックできます。それらがローカル・メモリーにある場合は、リダイレクターはそのセッション・スワップを防止することもできますし、あるいはその状況を処理するためのコードを呼び出すこともできます。

クライアントは、非同期APIの状態のために、セッション・スワップを防止する場合は、その前にQuery API Supportコールイン関数を呼び出して、そのAPIを処理するのに自分より適した他のクライアントがいないことをまず確認する必要があります。

Query Suspend関数呼び出しに失敗するクライアントがあると、そのクライアントを含め、すべてのクライアントがSession Activeコールアウト関数の呼び出しを受け取ることがあります。その結果、クライアントは先に対応するQuery SuspendあるいはSuspend Session呼び出しを受け取らなくてもSession Active呼び出しを受け取ることがあります。クライアントはこのSession Active呼び出しを無視してもかまいません。

Suspend Session（セッション保留）

このコールアウト関数はクライアントにセッション・スワップが実行されようとしていることを通知します。これが、クライアントがセッション・スワップを防止できる最後の機会となります。

例:

```
        CMP     AX,2                ;Suspend Session notification?
        JA      CHECK_NEXT_FUNCTION ;
        CMP     BX,OUR_SESSION_ID   ;BX has the current session ID

           ;ES:DI = The call-out address of the calling task-swapper
           ;Interrupts are disabled

        MOV     AX,SWAP_FLAG
        CMP     AX,0
        JE      RETURN_TO_CALLER

          ;Determine that the API is not being handled by another,
          ;  more competent client

        MOV     AX,6                ;Call-out to swapper
        MOV     BX,????             ;ID value of the asynchronous API
        CALL    OUT_TO_SWAPPER
        CMP     OFFSET API_INFO_STRUC,BX

          ;Does the most capable client have the same address as this
          ;  client?

        JNE     OTHER_CLIENT_TO_HANDLE
        MOV     AX,1
        JMP     RETURN_TO_CALLER

OTHER_CLIENT_TO_HANDLE:
        XOR     AX,AX

RETURN_TO_CALLER:
        RET

SWAP_FLAG  DB   0         ;Equals 0 if a session swap can be performed safely
                          ;Equals 1 if the client cannot safely handle a session
                          ;  swap.  All other values are reserved
                          ;
                          ;All other registers must be preserved.
CHECK_NEXT_FUNCTION:
```

解説

Query Suspend関数呼び出しに失敗するクライアントがない場合は、タスク・スワッパーは割り込みをディセーブルにし、Suspend Sessionコールアウト関数を呼び出します。これにより、クライアントはセッション・スワップを防止する最後の機会を得ることができます。ただし、クライアントはソフトウェア割り込みを出したり、割り込みをイネーブルにする呼び出しを行ったりすることはできません。

すべてのクライアントがAXをゼロにして戻ると、タスク・スワッパーは、割り込みをイネーブルにする前に、保管したコピーで、割り込みベクトル・テーブルを置き換えます。保管したコピーは、タスク・スワッパーが最初に起動したときのグローバル状態を示しています。これにより、保留されるセッションに対する割り込みハンドラーが、Suspend Session呼び出し時から、次にそのセッションが起動されるまで呼び出されないようにします。ローカル・ソフトウェアはSuspend Session呼び出しとActivate Session呼び出しとの間の割り込みを受け取ることはできません。これにより、ローカル・ソフトウェアがハードウェア割り込みに関して制御を獲得したり、グローバル・ソフトウェアが再開されたセッションのIDを受け取る前にグローバル・ソフトウェアに対して呼び出しを行ったりしないことが保証されます。ただし、グローバル・クライアントは、Suspend Session呼び出しの後から次のActivate Session呼び出しの間にも割り込みを受け取ることができます。この間、グローバル・ソフトウェアが非グローバル・メモリーの内容を仮定することはできません。Test Memory Regionコールアウト関数はメモリーのブロックをテストし、それがローカルであるかグローバルであるかを判定する必要があります。

クライアントは、非同期APIの状態のために、セッション・スワップを防止する場合は、その前にQuery API Supportコールイン関数を呼び出して、そのAPIを処理するのに自分より適した他のクライアントがいないことをまず確認する必要があります。Suspend Sessionコールイン関数呼び出しに失敗するクライアントがあると、そのクライアントを含め、すべてのクライアントがSession Activeコールイン関数の呼び出しを受け取ることがあります。その結果、クライアントは先に対応するQuery SessionあるいはSuspend Session呼び出しを受け取らなくても、Session Active呼び出しを受け取ることがあります。クライアントはこのSession Active呼び出しを無視してもかまいません。

Activate Session（セッション起動）

このコールアウト関数はクライアントに、セッションが活動状態になることを通知します。セッションが保留状態になっていた場合、それはメモリーに再導入され、そのローカルメモリーと割り込みベクトル・テーブルはスワップされる前の状態に戻されています。ただし、割り込みはディセーブルにされていて、このコールアウト関数の処理中もディセーブルのままでなければなりません。

例:

```
        CMP     AX,3               ;Activate Session call-in

          ;Interrupts are disabled and must remain disabled.
          ;Calls to DOS cannot be made

        JA      CHECK_NEXT_FUNCTION
        TEST    CX,0               ;If Bit 0 is set, indicates a new session
                                       ;  if not set, session was previously
                                       ;  suspended and is now being resumed.

        JNZ     TRACK_SESSION_IDS  ;If global client update list

          ;ES:DI = The call-out address of the calling task-swapper.

        XOR     AX,AX              ;for future extensibility

        RET                        ;All other registers are preserved.

TRACK_SESSION_IDS:
        MOV     [LIST_INDEX],BX    ;BX = ID of session being activated
        INC     LIST_INDEX
        INC     LIST_INDEX
        RET

CHECK_NEXT_FUNCTION:
```

解説

セッション・メモリーがスワップされている間、割り込みがイネーブルにされたとしても（そして、グローバル・ソフトウェアが割り込みを受け取り続けていても）、ローカル・ソフトウェアによって受け取られる割り込みはありません。ただし、セッションの割り込みベクトル・テーブルが再ロードされた後は、割り込みがイネーブルになるとすぐにハードウェア割り込みが起こり得ます。呼び出しが行われたときに割り込みがディセーブルでなければ、ローカル・ソフトウェアは割り込みを受け取り、グローバル・ソフトウェアへの呼び出しを行うことができます。しかし、そのソフトウェアは、新しいセッションIDを受け取っていないため、それを正しく処理できないかもしれません。
このセッションが初めて起動される、新しく作成されたセッションである場合は、

Create Sessionコールアウト関数呼び出しが、Activate Session呼び出しに優先します。

Session Active（セッション活動状態）

このコールアウト関数はクライアントに、セッションが活動状態になったことを通知します。このセッションが保留状態になっていた場合、それはメモリーに再導入され、そのローカル・メモリーおよび割り込みベクトル・テーブルはスワップされる前の状態に戻されています。

例:

```
        CMP     AX,4                ;Session Active call-in function
                                    ;Interrupts are enabled
                                    ;Calls to DOS can be made
        JA      CHECK_NEXT_FUNCTION

          ;ES:DI = call-out address of the calling task-swapper.

        TEST    CX,0                ;If Bit 0 is set, indicates a new session
                                    ;  if not set, session was previously
                                    ;  suspended and is now being resumed.
        JNZ     TRACK_SESSION_IDS   ;If global client update list

        XOR     AX,AX               ;for future extensibility

        RET                         ;All other registers are preserved.

TRACK_SESSION_IDS:
        MOV     [LIST_INDEX],BX     ;BX = ID of session being activated
        INC     LIST_INDEX
        INC     LIST_INDEX
        XOR     AX,AX               ;for future extensibility
        RET

CHECK_NEXT_FUNCTION:
```

解説

Query SuspendあるいはSuspend Sessionコールアウト関数呼び出しに失敗するクライアントがあると、そのクライアントを含め、すべてのクライアントがSession Activeコールアウト関数の呼び出しを受け取ることがあります。その結果、クライアントは、先に対応するQuery SuspendあるいはSuspend Session呼び出しを受け取っていなくても、Session Active呼び出し受け取ることがあります。クライアントはこのSession Active呼び出しを無視してもかまいません。

Create Session（セッション作成）

このコールアウト関数はクライアントに、タスク・スワッパーが新しいセッションを作成することを通知します。

例:

```
        CMP     AX,5            ;Create Session call-in function
                                ;Interrupts are enabled
                                ;Calls to DOS can be made
        JA      CHECK_NEXT_FUNCTION

          ;ES:DI = call-out address of the calling task-swapper.
          ;BX = The session ID of the new session.

        MOV     AX,CREATE_SESSION_FLAG
        RET

--------

CREATE_SESSION_FLAG   DB   0    ;=0 New Session can be created
                                ;=1 Client cannot handle a new session
                                ;All other values are reserved
```

解説

新しいセッションが作成されるとき、タスク・スワッパーはCreate Session関数を発行して、クライアントがセッションの作成を防止できるようにします。たとえば、固定長データ構造体内で各セッションの情報を保持しているようなグローバル・ソフトウェアは、この構造体に別のセッションの情報を格納する充分な余裕がなければ、この呼び出しを失敗させることができます。新しく作成されたセッションはすぐに起動するとは限りません。また、新しいセッションが活動状態になる前に、他のセッションが作成、破棄、スワップされるかもしれません。Create Sessionコールイン関数呼び出しに失敗するクライアントがあると、そのクライアントも含め、すべてのクライアントがDestroy Sessionコールイン関数の呼び出しを受け取ることがあります。その結果、クライアントは先に対応するCreate Session呼び出しがなくても、Destroy Session呼び出しを受け取ることがあります。クライアントはこのDestroy Session呼び出しを無視してもかまいません。

Destroy Session（セッション破棄）

この関数はクライアントに、タスク・スワッパーがセッションを破棄していることを通知します。

例:

```
        CMP     AX,6            ;Destroy Session call-in function?
                                ;Interrupts are enabled
                                ;Calls to DOS can be made
        JA      CHECK_NEXT_FUNCTION

          ;ES:DI = call-out address of the calling task-swapper
          ;BX = The session ID of the session being destroyed

        XOR     AX,AX           ;For future extensibility
        RET

                                ;All other registers are preserved.
```

解説

タスク・スワッパーは、セッションが破棄されているときに、Destroy Sessionコールアウト関数を呼び出します。通常、これは現行セッション内に適用業務が存在する場合に起こります。しかし、タスク・スワッパーを制御するセッション・マネージャーは、適用業務がまだ動作しているとき、あるいは保留状態の時にも、そのセッションを終了する方法をユーザーに提供することができます。そのため、破棄されているセッションは必ずしも現行セッションである必要はありません。Create Sessionコールイン関数呼び出しに失敗するクライアントがあると、そのクライアントを含め、すべてのクライアントがDestroy Sessionコールイン関数の呼び出しを受け取ることがあります。その結果、クライアントは先に対応するCreate Session呼び出しを受け取っていなくても、Destroy Session呼び出しを受け取ることがあります。クライアントはこのDestroy Session呼び出しを無視してもかまいません。

Swapper Exit（スワッパー終了）

このコールイン関数はグローバル・クライアントに、タスク・スワッパーがすでに活動状態ではないことを通知します。

例:

```
        CMP     AX,7      ;Notification task-swapper no longer active?
                          ;Interrupts are enabled
                          ;Calls to DOS can be made
        JA      OUT_OF_FUNCTIONS

          ;ES:DI = The call-out address of the calling task-swapper.


        TEST    OTHER_SWAPPER_PRESENT,BX  ;Other swapper present?
        XOR     AX,AX     ;AX = 0 for future extensibility
        RET


OTHER_SWAPPER_PRESENT  EQU     00000001B

          ; Bit 1 is set if no other active swappers
          ; Bit 1 is not set if at least one task-swapper
          ; remains after the calling task-swapper exits
          ; All other bits are reserved and must be 0

OUT_OF_FUNCTIONS:
```

解説

タスク・スワッパーが活動状態でなくなった場合に、タスク・スワッパーはこの関数を呼び出します。これにより、グローバル・ソフトウェアは、処理をディセーブルにし、タスク・スワッパーと共存するために必要な処理を実行することができます。

この関数は、タスク・スワッパー自体ではなく、実際のタスク・スワッパーを起動するソフトウェアにより呼び出すこともできます。このため、ES:DIで指定されたコールイン・アドレスは、別のコールアウト関数で渡されたアドレスと異なる場合もあり、また、NULLである場合もあります。

付録F.　ANSI.SYS（キーボード／画面の拡張制御）

ディスプレイのグラフィック属性やカーソルの動きを変更、制御したり、キーの再割り当てを行う機能を定義します。ANSI.SYSドライバーは、システムの画面とキーボードの制御を行う**ANSIエスケープ・シーケンス**の使用をサポートします。

書式

DEVICE=[ドライブ:] [パス] ANSI.SYS [/X]

パラメーター

[ドライブ:][パス]

ANSI.SYSドライバーのファイルが存在するドライブとパスを指定します。

スイッチ

/X　拡張キー（低位バイトがE0Hであるキー）が異なったキーとして再定義されることを許可します。拡張キーの再割り当ては、キーの再割り当てコマンドによって行われます。

解説

拡張キーの再割り当て

拡張キーボードを使用する場合、何個かの拡張キーを再割り当てしなおすため、/Xスイッチを使用する必要があるかもしれません。例えば、拡張キーボードには２つの&yms92.キーがあります。１つの&yms92.キーは数値キーのブロックにあり、もう１つはカーソル制御キーのブロックにあります。/Xスイッチを指定しない限り、この２つの&yms92.キーは同一のものとみなされます。

ANSIエスケープ・シーケンスによるカーソル移動、グラフィック、およびキーボードの設定

ANSIエスケープ・シーケンスはASCII文字の文字列で、その最初の２文字はエスケープ文字(1BH)と[記号(5BH)です。エスケープ文字(ESC)と[記号に続く文字または文字列は、キーボードまたはディスプレイの機能を制御する英数字コードを指定します。

次のANSIエスケープ・シーケンス一覧表では、略語ESCはASCIIエスケープ文字27(1BH)を表します（“ESC”という文字列ではありません）。エスケープ文字は、エスケープ・シーケンスの最初に置かれます。また、パラメーターの表記は以下のとおりです。

ANSIエスケープ・シーケンスで使用されるパラメーター

Pn　数値パラメーターです。10進数を指定します。

Ps　選択パラメーターです。機能を選択するため使用する10進数を指定します。パラメーターをセミコロン(;)で分離することにより２つ以上の機能を指定できます。

PL　行パラメーターです。ディスプレイまたはその他の装置の行の１つを表す10進数を指定します。

Pc　桁パラメーターです。画面またはその他の装置の桁の１つを表す10進数を指定します。

ANSIエスケープ・シーケンスの中で指定するすべての文字には、大文字と小文字の区別があります。

ESC[*PL*;*Pc*H

カーソル位置。

カーソルを指定された位置（座標）へ移動させます。位置を指定しなければ、カーソルはホーム・ポシション（行1, 桁1)へ移動します。このエスケープ・シーケンスは、次のカーソル位置エスケープ・シーケンス(ESC[*PL*;*Pc***f**)と同様に動作します。

例：`ESC[2;10H`

ESC[*PL*;*Pc*f

カーソル位置。

上記のカーソル位置エスケープ・シーケンス(ESC[*PL*;*Pc*H)と同様に動作します。

例：`ESC[4;10f`

ESC[*Pn*A

カーソルを上に移動。

桁を変更せずに指定された行数だけカーソルを上へ移動します。カーソルがすでに最上行にあるときは、ANSI.SYSドライバーはこのエスケープ・シーケンスを無視します。

例：`ESC[5A`

ESC[*Pn*B

カーソルを下に移動。

桁を変更せずに指定された行数だけカーソルを下へ移動します。カーソルがすでに最下行にあるときは、ANSI.SYSドライバーはこのエスケープ・シーケンスを無視します。

例：`ESC[10B`

ESC[*Pn*C

カーソルを右に移動。

行を変更せずに指定された桁数だけカーソルを右方へ移動します。カーソルがすでに右端の桁にあるときは、ANSI.SYSドライバーはこのエスケープ・シーケンスを無視します。

例：`ESC[4C`

ESC[*Pn*D

カーソルを左に移動。

行を変更せずに指定された桁数だけカーソルを左方へ移動します。カーソルがすでに左端の桁にあるときは、ANSI.SYSドライバーはこのエスケープ・シーケンスを無視します。

例：`ESC[4D`

ESC[s

カーソル位置の保存。

カーソルの現在位置を保存します。保存されたカーソル位置は、カーソル位置の復元エスケープ・シーケンス(ESC[**u**)を使用して復元できます。

ESC[u

カーソル位置の復元。

カーソル位置保存のエスケープ・シーケンス(ESC[**s**)によって保存された位置に、カーソルを移動します。

ESC[2J

画面の消去。

画面を消去し、カーソルをホーム・ポジション（行1, 桁1)へ移動します。

ESC[K

行の消去。

カーソル位置の文字を含めて、カーソル位置からその行の終わりまでのすべての文字を消去します。

ESC[*Ps*;...*Ps*m

グラフィック属性の設定。

次の値で指定されるグラフィック属性を制御します。指定された機能は、このエスケープ・シーケンスが次に現れるまで有効となります。

指定可能なグラフィック属性は、画面モードなどにより異なるため、以下に一覧を示します。表中、o印はその属性が有効、x印は無効であることを表します。

Ps	**テキスト属性**	**テキスト・モード**		**グラフィック・モード**	
		モノクロ*1	**カラー**	**モノクロ**	**カラー**
0	すべての属性を無効にする	o	o	o	o
1	高輝度（ボールド）	o	o	x	o
2	下線	o	x	x	x

Ps	**テキスト属性**	**テキスト・モード**		**グラフィック・モード**	
		モノクロ*1	**カラー**	**モノクロ**	**カラー**
5	明滅*2	o	o	x	x
7	反転表示	o	o	x	x
8	取り消し（非表示）	o	o	x	o
30	黒の前景色	x	o	x	o
31	赤の前景色	x	o	x	o
32	緑の前景色	x	o	x	o
33	黄の前景色	x	o	x	o
34	青の前景色	x	o	x	o
35	紫の前景色	x	o	x	o
36	水色の前景色	x	o	x	o
37	白の前景色	x	o	x	o
40	黒の背景色	x	o	x	x
41	赤の背景色	x	o	x	x
42	緑の背景色	x	o	x	x
43	黄の背景色	x	o	x	x
44	青の背景色	x	o	x	x
45	紫の背景色	x	o	x	x
46	水色の背景色	x	o	x	x
47	白の背景色	x	o	x	x

注:

*1: 日本語モードには、モノクロのテキスト・モードはありません。

*2: 英語モードでのみサポートされます。

30から47の範囲の値はISO6429規格に準拠しています。

ESC[=*Ps*h

モードの設定。

画面の表示幅やカラー機能を、次の値のどれか１つで指定されたモードに変更します。

0	40文字×25行	モノクロ（テキスト）
1	40文字×25行	カラー（テキスト）
2	80文字×25行	モノクロ（テキスト）
3	80文字×25行	カラー（テキスト）
4	320×200ドット	4色カラー（グラフィック）
5	320×200ドット	モノクロ（グラフィック）
6	640×200ドット	モノクロ（グラフィック）
7	ラップアラウンド処理を許可する	
13	320×200ドット	カラー（16色グラフィック）
14	640×200ドット	カラー（16色グラフィック）

15	640×350ドット	モノクロ（2色グラフィック）
16	640×350ドット	カラー(16色グラフィック）
17	640×480ドット	モノクロ(2色グラフィック）
18	640×480ドット	カラー(16色グラフィック）
19	320×200ドット	カラー(256色グラフィック）
114	640×480ドット	カラー(16色グラフィック：80文字×25行）
115	80文字×25行	カラー(拡張テキスト）

注:　日本語モードでは、3, 17, 18, 114または115が有効です。英語モードでは、114と115は意味をもちません。

ESC[=*Ps*l

モードのリセット。

モードをリセットします。指定できる値は、モードの設定と同じです。ただし、７は、ラップアラウンド処理を禁止します。このエスケープ・シーケンスの最後の文字は小文字のＬです。

ESC[コード;文字列p

キーの再割り当て。

キーボードのキーを特定の文字列として再定義します。このエスケープ・シーケンスのパラメーターは次のように定義されます。

- コードは次の表に示された値です（２つの値で表すものもあります）。これらの値は、キーボードのキーとキーの組み合わせを表します。コマンドの中にこれらの値を使用するときは、エスケープ・シーケンスに必要なセミコロン(;)に加えて、下記の表に示されているセミコロン(;)も入力しなければなりません。かっこ（　）内のコードは、キーボードによっては利用できません。ANSI.SYSドライバーに対するDEVICEコマンドで/Xスイッチを指定しない限り、ANSI.SYSドライバーはかっこ（　）内のコードを解釈しません。

- 文字列は１文字分のASCIIコード、または引用符(")で囲まれた文字列です。例えば、65と"A"はどちらも大文字のＡを表します。

注:　キー・コードは、キーボードの種類により若干異なります。次の表は、5576-A01型キーボードについて記述しています。詳しくは、**「DOS/Vプログラミング解説編」**を参照してください。

表　F-1 (1/5). ASCIIキー・コード一覧

キー	コード	⇧ + コード	Ctrl + コード	Alt + コード
F1	0;59	0;84	0;94	0;104
F2	0;60	0;85	0;95	0;105
F3	0;61	0;86	0;96	0;106
F4	0;62	0;87	0;97	0;107

表　F-1 (2/5). ASCIIキー・コード一覧

キー	コード	Shift + コード	Ctrl + コード	Alt + コード
F5	0;63	0;88	0;98	0;108
F6	0;64	0;89	0;99	0;109
F7	0;65	0;90	0;100	0;110
f8	0;66	0;91	0;101	0;111
F9	0;67	0;92	0;102	0;112
F10	0;68	0;93	0;103	0;113
F11	(0;133)	(0;135)	(0;137)	(0;139)
F12	(0;134)	(0;136)	(0;138)	(0;140)
Home （数値キー）	0;71	55	0;119	---
↑ （数値キー）	0;72	56	(0;141)	---
Page Up （数値キー）	0;73	57	0;132	---
← （数値キー）	0;75	52	0;115	---
→ （数値キー）	0;77	54	0;116	---
End （数値キー）	0;79	49	0;117	---
↓ （数値キー）	0;80	50	(0;145)	---
Page Down （数値キー）	0;81	51	0;118	---
Insert （数値キー）	0;82	48	(0;146)	---
Delete （数値キー）	0;83	46	(0;147)	---
Home	0;71 (224;71)	0;71 (224;71)	0;119 (224;119)	(224;151)
↑	0;72 (224;72)	0;72 (224;72)	(224;141)	(224;152)
PgUp	0;73 (224;73)	0;73 (224;73)	0;132 (224;132)	(224;153)
←	0;75 (224;75)	0;75 (224;75)	0;115 (224;115)	(224;155)
→	0;77 (224;77)	0;77 (224;77)	0;116 (224;116)	(224;157)
End	0;79 (224;79)	0;79 (224;79)	0;117 (224;117)	(224;159)
↓	0;80 (224;80)	0;80 (224;80)	(224;145)	(224;154)
PgDn	0;81 (224;81)	0;81 (224;81)	0;118 (224;118)	(224;161)
Ins	0;82 (224;82)	0;82 (224;82)	(224;146)	(224;162)
Del	0;83 (224;83)	0;83 (224;83)	(224;147)	(224;163)

表　F-1 (3/5). ASCIIキー・コード一覧

キー	コード	Shift + コード	Ctrl + コード	Alt + コード
Print Screen	---	---	0;114	---
Pause	---	---	0;0	---
Back Space	8	8	127	(0)
Enter	13	13	10	(0;28)
Tab	9	0;15	(0;148)	(0;165)
Esc	27	27	27	(0;01)
A	97	65	1	0;30
B	98	66	2	0;48
C	99	67	3	0;46
D	100	68	4	0;32
E	101	69	5	0;18
F	102	70	6	0;33
G	103	71	7	0;34
H	104	72	8	0;35
I	105	73	9	0;23
J	106	74	10	0;36
K	107	75	11	0;37
L	108	76	12	0;38
M	109	77	13	0;50
N	110	78	14	0;49
O	111	79	15	0;24
P	112	80	16	0;25
Q	113	81	17	0;16
R	114	82	18	0;19
S	115	83	19	0;31
T	116	84	20	0;20
U	117	85	21	0;22
V	118	86	22	0;47
W	119	87	23	0;17

表　F-1 (4/5). ASCIIキー・コード一覧

キー	コード	Shift + コード	Ctrl + コード	Alt + コード
X	120	88	24	0;45
Y	121	89	25	0;21
Z	122	90	26	0;44
1	49	33	---	0;120
2	50	34	0;3	0;121
3	51	35	---	0;122
4	52	36	---	0;123
5	53	37	---	0;124
6	54	38	30	0;125
7	55	39	---	0;126
8	56	40	---	0;126
9	57	41	---	0;127
0	48	(0;11)	---	0;129
-	45	61	31	0;130
^	94	126	---	0;131
[	91	123	27	(0;27)
]	93	125	29	(0;43)
:	58	42	---	(0;40)
;	59	43	---	0;39
@	64	96	---	(0;26)
,	44	60	---	(0;51)
.	46	62	---	(0;52)
/	47	63	---	(0;53)
¥	92	124	28	---
＼	92	95	28	---
Enter（数値キー）	13 (224;13)	13 (224;13)	10 (224;13)	(0;166)
/（数値キー）	47 (224;47)	47 (224;47)	(0;149)	(0;164)
*（数値キー）	42	42	(0;150)	(0;55)

表　F-1 (5/5). ASCIIキー・コード一覧

キー	コード	Shift + コード	Ctrl + コード	Alt + コード
- （数値キー）	45	45	(0;142)	(0;74)
+ （数値キー）	43	43	(0;144)	(0;78)
5 （数値キー）	(0;76)	53	(0;143)	---

例

- 文字列を使用して、円記号(¥)キーと疑問符(?)キーを入れ換えるには、次のエスケープ・シーケンスを入力してください。

```
ESC["¥";"?"pESC["?";"¥"p
```

- 各キーのASCIIコードを使用して、円記号(¥)キーと疑問符(?)キーを入れ換えるには、次のエスケープ・シーケンスを入力してください。

```
ESC[92;63pESC[63;92p
```

- 円記号(¥)キーと疑問符(?)キーの元の機能を回復するには、次のエスケープ・シーケンスを入力してください。

```
ESC[92;92pESC[63;63p
```

テクニカル・リファレンス
DOS/V BIOS インターフェース
技術解説編

目次

図

表

特記事項

「DOS/V BIOSインターフェース技術解説編」は、インターナショナル・ビジネス・マシーンズ・コーポレーション（以下IBMという）の著作物です。これはIBMの出版物「IBM DOSバージョンJ5.0/V BIOSインターフェース技術解説書」の内容を記載しており、これらの著作権はIBMが所有しています。ただし、本文中の20ページから21ページ(INT 10H, AH=12H)の内容は、OADG協議会が所有しています。

これらの内容の無断転載・複製などの著作権に抵触する使用はできません。

本書で使用されているIBMは、米国IBM社の商標です。

本文中のESC/Pは、セイコーエプソン株式会社の登録商標です。

第1章　はじめに

本章では、割り込みの詳細について説明します。

割り込みベクトル

80x86系処理装置のINT命令は、絶対アドレス0から始まる割り込みベクトル・テーブル中に記憶されている絶対アドレスを呼び出します。その際、INT命令のオペランドとして指定された割り込みタイプに対応するベクトル（絶対アドレス=4×割り込みタイプ）が選ばれます。

この割り込みベクトルのうち、いくつかはBIOSおよびDOSのルーチンのアドレスを含んでいるので、ユーザー・プログラムは、適当な割り込みタイプを指定したINT命令を発行することによって、これらのルーチンを呼び出すことができます。この節では、利用できる割り込みタイプとその機能について説明します。

以下に割り込みベクトル・テーブルを示します。

表 1. 割り込みベクトル・テーブル（２の１）

割り込みタイプ		名前
0		0による除算
1		シングル・ステップ割り込み
2		マスク不可割り込み
3		ブレーク・ポイント割り込み
4		オーバーフロー
5		画面印刷
6, 7		予約済み
8	8259 NO.1 割り込み ベクトル	タイマー割り込み
9		キーボード割り込み
A		H/W割り込み
B		H/W割り込み
C		H/W割り込み
D		予約済み
E		フロッピーディスク割り込み
F		H/W割り込み

表 2. 割り込みベクトル・テーブル（2の2）		
割り込みタイプ		**名前**
10 11 12 13 14 15 16 17 18 19 1A	BIOS 入口点	ディスプレイ入出力 装置構成情報 記憶域サイズの情報 ディスク・ドライブ入出力 ASYNC通信入出力 システム入出力 キーボード入出力 プリンター入出力 予約済み 予約済み 時刻
1B 1C	ユーザー が用意	キーボードBreakアドレス タイマー刻み
1D, 1E, 1F		予約済み
20 21 22 23 24 25 26 27 28 29～2E 2F 30～3F 40～5F 60～66 67 68～6F		プログラム終了 DOSファンクション・コール 終了アドレス Break出口アドレス 致命的エラー処理ルーチン・ベクトル 絶対ディスク読取り 絶対ディスク書込み 終了後、常駐 DOSが内部的に使用 予約済み 多重割り込み 予約済み 予約済み ユーザー・プログラムで使用可 EMS 予約済み

表 3. 割り込みベクトル・テーブル（２の３）

割り込みタイプ		名前
70 71 72 73 74 75 76 77	8259 NO.2 割り込み ベクトル	リアル・タイム・クロック割り込み 予約済み H/W割り込み H/W割り込み マウス割り込み 数値演算プロセッサー割り込み ハードディスク割り込み H/W割り込み
78～FF		予約済み

第2章　BIOS割り込み

ユーザー・プログラムから、タイプ10H～1AHの割り込みによって直接（つまりDOSを経由しないで）BIOSのルーチンを呼び出し、低レベルのブロック（ディスク）または文字（ディスプレイ、ASYNC通信ポート、キーボードおよびプリンター）入出力や装置操作を行うことができます。ただし、これらのルーチンは、ハードウェア依存性が高く、正しく使用するためには個々の装置に対する知識を必要とします。したがって、代替できる場合は、DOS機能呼出しを用いることをお勧めします。

BIOSルーチンに渡される、またはBIOSルーチンから返されるすべてのパラメーターは、CPUのレジスターによって受け渡しされます。使用されるレジスターは、以下の個々のルーチンの項で説明します。

1つのBIOSルーチンがいくつかの異なった機能を持っている場合は、希望する機能の番号をAHレジスターに設定して呼び出します。

一般に、BIOSルーチンはAXとフラグ以外のすべてのレジスターを保持します。他のレジスターは、そのレジスターを使って呼出し元に値を返すときにだけ変更されます。詳細は、個々のルーチンの説明を参照してください。

定義されているBIOS割り込みを次ページ以降に示します。見出しに表示される割り込みタイプはすべて16進で表わされています。

INT 10H ディスプレイ入出力

AH=00H ビデオ・モードの設定

＜設定＞

(AL) 設定するビデオ・モード

ALの値に従って、表示画面のモードを次のように設定します。

表 4. 日本語モードの表示画面の設定

AL (16進数)	モード	表示文字	文字ボックス		画面サイズ
			全角	半角	
00~02	予約済み	—	—	—	—
03	16色文字モード*1	80×25	16×19	8×19	640×475
04~10	予約済み	—	—	—	—
11	2色カラー・グラフィック*2	80×30	16×16	8×16	640×480
12	16色カラー・グラフィック*2	80×30	16×16	8×16	640×480
13	予約済み	—	—	—	—
72	16色カラー・グラフィック*2	80×25	16×19	8×19	640×480
73	16色文字モード	80×25	16×19	8×19	640×475

注:

***1** このモードは初期モードです。

***2** ドットの表示色はパレット番号で指定します（パレットの初期値については『カラー値と表示色の関係』の項を参照）。

ビデオ・モードについての補足説明

モード 11H

この画面モードは、英語モードの画面モード11Hと互換性があります。文字のイメージはビデオ・バッファーにマップされますが、文字属性は以下のように処理されます。

ビット	意味
7	ビット7がオンの場合、文字のイメージ・データは現行のビデオ・バッファーとXOR（排他的論理和）されます。

0~6	オン・オフの値に関係なく無視されます。

モード 12H

この画面モードは、英語モードの画面モード12Hと互換性があります。文字属性は前景色（パレット番号を指定）を指定するために使用します。属性の前景色に従って文字イメージがビデオ・バッファーにマップされます。文字属性は以下の意味を持ちます。

ビット	意味
7	ビット7がオンの場合、文字のイメージ・データは、現行のビデオ・バッファーとXOR（排他的論理和）されます。
4~6	オン・オフの値に関係なく無視されます。
0~3	文字の色をパレット番号で指定します（0~15）。

モード 03H

モード03H（エミュレートCGA（=カラー・グラフィックス・アダプター）テキスト・モード）はINT10Hインターフェースの観点から見ると英語モードのモード03H（CGAテキスト・モード）と機能的に同じです。文字属性バイトは、前景色と背景色を指定します。このモードは、VGA（=ビデオ・グラフィックス・アレイ）の16色グラフィック・モードを使用して実現しているため、実テキスト・バッファーは存在しません。文字は、INT10Hインターフェースを通じてのみ表示されます。文字属性バイトは以下の意味を持ちます。

ビット	意味
7	明滅／背景色の高輝度（ただし、ハードウェアが明滅をサポートしていない場合、背景色の高輝度としてのみ機能します）
6	背景色の赤
5	背景色の緑
4	背景色の青
3	前景色の高輝度
2	前景色の赤
1	前景色の緑
0	前景色の青

このモードでは、ビデオ・バッファーがアプリケーション・プログラムにより直接アクセスされていなければ、画面イメージは文字コード／属性の組み合わせで完全に保管と復元が行われます。上下方向の画面移動（スクロール）はCRTCレジスターのAPA開始アドレスを変更することで、高速に行われます。

モード 73H

モード73H（エミュレート拡張CGAテキスト・モード）は、以下の文字属性を指定することができます。属性バイトの1と2は「文字ブロックの読み取り(INT10H, AX=131xH)」と「文字ブロックの書き込み（INT10H, AX=132xH)」の機能により読み出しおよび書き込みを行うことができます。この２つの機能を使用しない場合は、このモードは、モード03Hと同じ働きをすることになります。

このモードでは、次の３とおりの文字の書き込みが利用できます。

- 文字コードのみ

- 文字コードと属性バイト0
- 文字コードと属性バイト0, 1, 2

文字コードのみの場合、現在の文字属性を変更せずに、文字を書き込みます。文字コードと属性バイト0の場合、バイト0に指定した属性とバイト1と2の省略時解釈値の属性(どちらも00H)を用いて文字を書き込みます。文字コードと属性バイト0, 1, 2の場合、バイト0, 1, 2に指定した属性に従って文字を書き込みます。

属性バイト　0

ビット	意味
7	背景色の高輝度
6	背景色の赤
5	背景色の緑
4	背景色の青
3	前景色の高輝度
2	前景色の赤
1	前景色の緑
0	前高色の青

属性バイト　1

ビット	意味
7	下線
6	反転用に予約済み（0に設定）
5	明滅用に予約済み（0に設定)
4	透過属性用に予約済み（0に設定)
3	左罫線
2	横罫線
1	予約済み（0に設定)
0	予約済み（0に設定)

属性バイト　2

ビット	意味
0~7	予約済み（0に設定)

注:

1. 罫線は、白色(パレット番号7)で描かれます。
2. 下線は、現在の文字の前景色で描かれます。

モード 72H

この画面モードは、表示文字(80×25)と文字ボックス(8×19)を除いて画面モード12Hと同じです。文字属性バイトは、文字の色（パレット番号）を指定します。文字イメージは、属性でマップされます。属性は以下の意味を持ちます。

ビット	意味
7	オンの場合、現在のビデオ・バッファーのドットとXORされます。
4~6	オン・オフに関係なく無視されます。
0~3	文字の色をパレット番号で指定します。

AH=01H カーソル・タイプの設定

カーソルは文字モードの場合のみ表示されます。グラフィック・モードの場合は、アプリケーション・プログラムによってカーソルを独自にAPAバッファーに描く必要があります。カーソルの形（タイプ）はCXの各ビットに設定した値に従って決まります。カーソルは明滅できませんし、全角幅の表示をサポートしていません。

＜設定＞

(CH)　文字ボックス内のカーソル開始行
(CL)　文字ボックス内のカーソル終了行

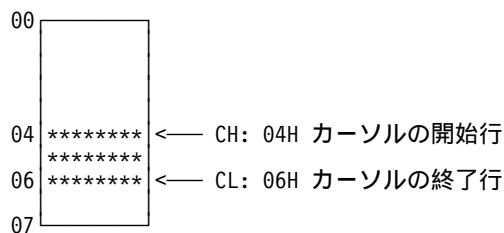
次のようにカーソル・タイプが設定されます。

カーソル・タイプの推奨値を次に示します。カーソルの実際の形状については、図1を参照してください。

表 5. カーソルの形状と設定値

カーソルの形状	カーソル開始行（16進）CH	カーソル終了行（16進）CL
底形	06	07
上部	00	03
下部	04	07
箱形(全部)	00	07
カーソル・オフ	20	00

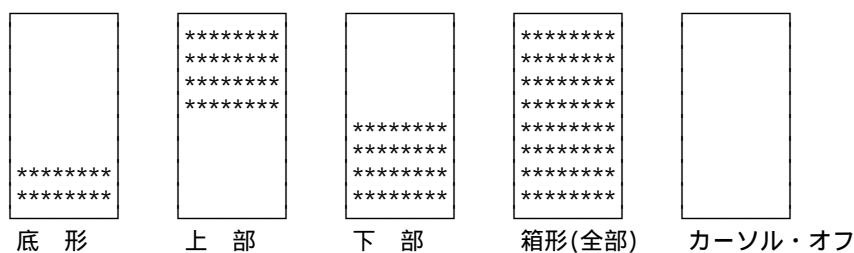



図 1. カーソルの形状

注:

1. 底形が省略値のカーソル・タイプです。
2. CXのビット13がONの場合、カーソルは表示されません（つまり、CH=20の場合）。

AH=02H カーソル位置の設定

＜設定＞

(BH) ページ番号（0に設定）

(DH, DL) カーソルを設定する位置（行番号、桁番号）を指定します。桁番号は半角で数えます。（0, 0）が左上隅を意味します。

AH=03H カーソル位置の読み取り

現在のカーソル位置（行番号、桁番号）とカーソル・タイプが、それぞれ読み取られます。

＜設定＞

(BH) ページ番号（0に設定）

＜結果＞

(DH,DL) カーソル位置の行番号、桁番号

(CH,CL) カーソル・タイプ（つまり、カーソルの開始行、終了行）

AH=04H 予約済み

AH=05H 活動ページの選択

活動ページをALに設定します（0から始まる）。活動ページの内容がディスプレイに表示されます。

＜設定＞

(AL) 活動ページ番号（0に設定）

日本語モードにおける活動ページは、0のみです。0を指定してください。

AH=06H　上方向に画面移動

上方向に移動したい画面の範囲をCXとDXで指定し、移動する行数をALに指定します。移動により下部に埋められるブランク行の属性をBHに指定します。

＜設定＞

(AL)　移動する行数を指定しますAL=00Hの場合、CXとDXで指定した範囲の画面がブランクになります。

(CH,CL)　左上隅（行番号、桁番号）を指定

(DH,DL)　右下隅（行番号、桁番号）を指定

(BH)　移動により下部に埋められるブランク行の属性

AH=07H　下方向に画面移動

指定した範囲の画面を下方向に画面移動します。

＜設定＞

(AL)　移動する行数を指定しますAL=00Hの場合、CXとDXで指定した範囲の画面がブランクになります。

(CH,CL)　左上隅（行番号、桁番号）

(DH,DL)　右下隅（行番号、桁番号）

(BH)　移動により上部に埋められるブランク行の属性

AH=08H　現在のカーソル位置の属性と文字の読み取り

文字がALに、属性がAHに、それぞれ読み取られます。

カーソルが全角文字の左側に位置していた場合、その全角文字の第１バイト目が返され、右側に位置していた場合、第２バイト目が返されます。

＜設定＞

(BH)　ページ番号（0に設定）

＜結果＞

(AL)　カーソル位置の文字コード

(AH)　カーソル位置の文字の属性（文字モードのみ）

この機能はグラフィック・モードにおいても有効です。

AH=09H　現在のカーソル位置への属性と文字の書き込み

書き込む文字をALに、その文字の属性をBLに、書き込む文字数を半角単位でCXに、それぞれ指定します。

＜設定＞

(BH)　ページ番号（0に設定）

(AL)　文字コード

(CX)　書き込む文字数

(BL)　文字の属性

全角文字を書き込む場合は、CXに1を指定して第1バイトと第2バイトを続けて書き込むようにしてください。第1バイト目を書き込んだ後、カーソルを次の桁に移動して第2バイト目を書き込みます。第1バイトと第2バイトを書く間に、他の命令を実行した場合、その結果は保証の限りではありません。ただし、'INT10H, AH=13H 文字列の書き込み' 命令は例外です。

注: AH=09HとAH=0AHの機能は類似していますが、グラフィック・モードの場合は、AH=09Hを使用してください。

AH=0AH 文字のみの書き込み

属性が書き込まれない（したがってBLには何も指定しない）ことを除けば、AH=09Hと同じです。

＜設定＞

(BH) ページ番号（0に設定）
(AL) 文字コード
(CX) 書き込む文字数

注: グラフィック・モードの場合、AH=09Hを使用してください。

AH=0BH 予約済み

AH=0CH ドットの書き込み（グラフィック・モードでのみ有効）

<設定>

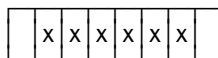
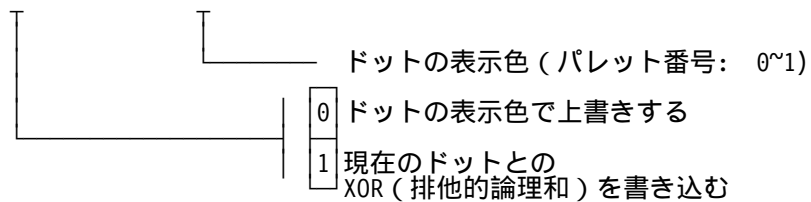
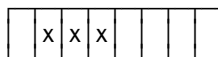
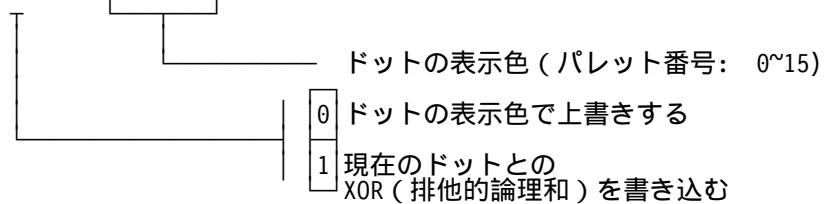
(BH) ページ番号（0に設定）

(DX) 行番号（ドット単位）（画面上端が0）

(CX) 桁番号（ドット単位）（画面左端が0）

(AL) カラー値

```
 7 6 5 4 3 2 1 0
┌─┬─┬─┬─┬─┬─┬─┬─┐
│ │x│x│x│x│x│x│ │   ２色グラフィック・モード
└─┴─┴─┴─┴─┴─┴─┴─┘
 │           └──── ドットの表示色（パレット番号: 0~1)
 │          │0│ ドットの表示色で上書きする
 └──────────┤
            │1│ 現在のドットとの
                 XOR（排他的論理和）を書き込む
```

```
 7 6 5 4 3 2 1 0
┌─┬─┬─┬─┬─┬─┬─┬─┐
│ │x│x│x│ │ │ │ │   16色カラー・グラフィック・モード
└─┴─┴─┴─┴─┴─┴─┴─┘
 │       └──┬──┘
 │          └──── ドットの表示色（パレット番号: 0~15)
 │          │0│ ドットの表示色で上書きする
 └──────────┤
            │1│ 現在のドットとの
                 XOR（排他的論理和）を書き込む
```

カラー値と表示色の関係

16色カラー・グラフィック・モード

カラー値（4ビット）を指定することにより16個（2^4）のパレットから任意の1個が選択されます。各パレットには対応するカラー・レジスター番号（0~63）が設定されており、64個のカラー・レジスターの中から任意の1個が選択されます。各カラー・レジスターには赤、緑、青の各色要素について各6ビットの明度（2^6=64階調）が設定されており、この設定データにより最終的な表示色が決まります。

図 2. カラー値と表示色の関係

パレットの初期値（16色カラー・グラフィック・モード）

表 6. パレットの初期値

パレット番号	カラー・レジスター番号／表示色	パレット番号	カラー・レジスター番号／表示色
0	0（黒）	8	56（灰色）
1	1（青）	9	57（薄い青）
2	2（緑）	10	58（薄い緑）
3	3（水色）	11	59（薄い水色）
4	4（赤）	12	60（薄い赤）
5	5（紫）	13	61（薄い紫）
6	20（茶色）	14	62（薄い黄）
7	7（白）	15	63（明るい白）

AH=0DH　ドットの読み取り（グラフィック・モードでのみ有効）

＜設定＞

(DX)　　行番号（ドット単位）（画面上端が0）

(CX)　　桁番号（ドット単位）（画面左端が0）

＜結果＞

(AL)

AH=0EH テレタイプ式書き込み

これは現在のカーソル位置に1文字を書き込み、カーソルを1つ進めます。カーソルが（最下-1）行目の最後の桁にある場合、書き込み後に1行上方に画面移動して、新しい（最下-1）行目の先頭にカーソルを進めます。
最下行目は、かな漢字変換とキーボード・シフト状況の標識域に使用されているので、画面移動の範囲には入りません。

全角文字を各行の最後の桁（第79桁）に書き込もうとすると、その桁には半角ブランクが書き込まれ、次行の最初にその全角文字が書き込まれます。

この機能はグラフィック・モードにおいても有効です。

＜設定＞

(AL) 書き込む文字を指定します

(BL) 文字の色を指定します（グラフィック・モードの場合に必要で、前景色を指定します）

注:

1. 復帰、改行、後退、およびベルの各文字は、制御コードとして取り扱われ、画面に表示されません。
2. 画面が1行上方向に移動するときに最下行に埋められるブランク行の属性は、文字モードの場合、上方向に移動する前の最下行の最初の桁の属性によって決まります。グラフィック・モードの場合は、最下行に埋められるブランク行の属性は、常に黒(00H)です。

AH=0FH 現在のディスプレイ状況

次の情報が得られます。

＜結果＞

(AL) 現在設定されているビデオ・モード

(AH) 画面上の文字桁の数

(BH) 常に0が返されます。

AH=10H パレット・レジスターの設定

(AL) = 00H パレット個別設定

＜設定＞

(BL) 設定するパレット・レジスターの番号(0~15)

(BH) パレット・レジスターに設定するカラー・レジスターの番号を指定します（各カラー・レジスターのカラー値は、『省略時のカラー・レジスター構成と表示色』を参照）。

(AL) = 01H オーバー・スキャン設定

＜設定＞

(BH) オーバー・スキャン・レジスターに設定するカラー・レジスターの番号

(AL) = 02H パレット一括設定

＜設定＞

(ES:DX) パレット・レジスター(0~15) とオーバースキャン・レジスターに設定する値が置かれている記憶域のテーブルの先頭アドレスを指定します。

テーブルの内容

バイト0~15	パレット・レジスター0~15への設定値
バイト16	オーバースキャン・レジスターへの設定値

(AL) = 07H パレット個別読み取り

＜設定＞

(BL) 読み取るパレット・レジスターの番号（0~15)

＜結果＞

(BH) 指定したパレット・レジスターに設定されているカラー・レジスターの番号

(AL) = 08H オーバー・スキャン・レジスター読み取り

＜結果＞

(BH) 読み取られたオーバー・スキャン・レジスターの値

(AL) = 09H パレット一括読み取り

＜設定＞

(ES:DX) 読み取るパレット・レジスター（0~15）とオーバースキャン・レジスターの値を置く記憶域の先頭アドレスを指定します。

＜結果＞（ES:DX）で指定した記憶域に以下の内容が返されます。

バイト0~15	読み取られたパレット・レジスター0~15の値
バイト16	読み取られたオーバースキャン・レジスターの値

(AL) = 10H カラー・レジスター個別設定

＜設定＞

(BX) 設定するカラー・レジスターの番号

(DH) 設定する赤色の値

(CH) 設定する緑色の値

(CL) 設定する青色の値

(AL) = 12H カラー・レジスターの一括設定

＜設定＞

(ES:DX) 色が設定されているテーブルの先頭アドレスを指定します。テーブルの形式: 赤、緑、青、赤、緑、青、...

(BX) 設定する最初のカラー・レジスターの番号

(CX) 設定するカラー・レジスターの数

(AL) = 15H カラー・レジスターの個別読み取り

＜設定＞

(BX) 読み取るカラー・レジスターの番号

＜結果＞

(DH) 読み取った赤色の値

(CH) 読み取った緑色の値

(CL) 読み取った青色の値

(AL) = 17H カラー・レジスターの一括読み取り

＜設定＞

(ES:DX) 読み取った値を書き込むテーブルの先頭アドレスを指定します。

テーブルの形式: 赤、緑、青、赤、緑、青、...

(BX) 読み取る最初のカラー・レジスターの番号

(CX) 読み取るカラー・レジスターの数

＜結果＞

(ES:DX) 読み取られた値が入っているテーブルの先頭アドレス

(AL) = 他の値 予約済み

省略時のカラー・レジスター構成と表示色

省略時のカラー・レジスターの値とその表示色は、以下のとおりです。

表 7. 16色カラー・グラフィック・モードのカラー・レジスター値

番号	-赤 -緑 -青 表示色	番号	-赤 -緑 -青 表示色	番号	-赤 -緑 -青 表示色
0	00H 00H 00H 黒	1	00H 00H 2AH 青	2	00H 2AH 00H 緑
3	00H 2AH 2AH 水色	4	2AH 00H 00H 赤	5	2AH 00H 2AH 紫
6	2AH 2AH 00H 黄	7	2AH 2AH 2AH 白	8	00H 00H 15H 暗い青
9	00H 00H 3FH 明るい青	10	00H 2AH 15H	11	00H 2AH 3FH 緑青
12	2AH 00H 15H	13	2AH 00H 3FH	14	2AH 2AH 15H
15	2AH 2AH 3FH	16	00H 15H 00H 暗い緑	17	00H 15H 2AH
18	00H 3FH 00H 明るい緑	19	00H 3FH 2AH 空色		
20	2AH 15H 00H 茶色	21	2AH 15H 2AH	22	2AH 3FH 00H
23	2AH 3FH 2AH	24	00H 15H 15H 暗い水色	25	00H 15H 3FH
26	00H 3FH 15H 青緑	27	00H 3FH 3FH 明るい水色	28	2AH 15H 15H
29	2AH 15H 3FH	30	2AH 3FH 15H	31	2AH 3FH 3FH
32	15H 00H 00H 暗い赤	33	15H 00H 2AH	34	15H 2AH 00H 黄緑
35	15H 2AH 2AH	36	3FH 00H 00H 明るい赤	37	3FH 00H 2AH 濃いピンク
38	3FH 2AH 00H やまぶき色	39	3FH 2AH 2AH	40	15H 00H 15H 暗い紫
41	15H 00H 3FH すみれ色	42	15H 2AH 15H	43	15H 2AH 3FH
44	3FH 00H 15H 紫赤	45	3FH 00H 3FH 明るい紫	46	3FH 2AH 15H
47	3FH 2AH 3FH	48	15H 15H 00H 暗い黄	49	15H 15H 2AH
50	15H 3FH 00H	51	15H 3FH 2AH	52	3FH 15H 00H
53	3FH 15H 2AH	54	3FH 3FH 00H 明るい黄	55	3FH 3FH 2AH
56	15H 15H 15H 灰色	57	15H 15H 3FH 薄い青	58	15H 3FH 15H 薄い緑
59	15H 3FH 3FH 薄い水色	60	3FH 15H 15H 薄い赤	61	3FH 15H 3FH 薄い紫
62	3FH 3FH 15H 薄い黄	63	3FH 3FH 3FH 明るい白		

カラー・レジスター番号は10進値です。

AH=11H 文字の生成

ユーザー独自の１バイト文字のフォント・セットを設定することができます。

(AL) =00H ユーザーの１バイト文字セットの設定

＜設定＞

(ES:BP) 記憶域のユーザー・テーブルへのポインター
(CX) 設定する文字数
(DX) テーブル内の最初の文字の文字コード
(BL) 設定するブロック（0に設定）
(BH) 文字当たりのバイト数

(AL) =他の値 予約済み

AH=12H 機能の選択

(BL) = 20H プリント・スクリーン処理の切替

注: INT 05Hの選択をします。

(BL) = 31H デフォルト・パレットのロード

＜設定＞
(AL) = 00H デフォルト・パレット・ローディングのイネーブル
(AL) = 01H デフォルト・パレット・ローディングのディセーブル
＜結果＞
(AL) = 12H この機能がサポートされている

注: この機能をディセーブルにすると、次のビデオ・モードの設定でデフォルト・パレットが設定されません。

(BL) = 32H ビデオ

＜設定＞
(AL) = 00H ビデオのイネーブル
(AL) = 01H ビデオのディセーブル
＜結果＞
(AL) = 12H この機能がサポートされている

注: この機能をディセーブルにすると画面表示は更新されず、CPUはビデオのI/Oポート、VRAMへのアクセスができなくなります。

(BL) = 33H グレースケールの設定

＜設定＞
(AL) = 00H グレースケールのイネーブル
(AL) = 01H グレースケールのディセーブル
＜結果＞
(AL) = 12H この機能がサポートされている

注: この機能をイネーブルにすると、次のビデオ・モードの設定(AH=00H), もしくはパレット・レジスタの設定(AH=10H)ファンクション実行後、グレースケールで表示されます。

(BL) = 34H カーソル・エミュレーション

＜設定＞
(AL) = 00H カーソル・エミュレーションのイネーブル
(AL) = 01H カーソル・エミュレーションのディセーブル
＜結果＞
(AL) = 12H この機能がサポートされている

注: この機能をイネーブル（パワーオン時の既定値はイネーブル）にするとカーソル形状設定(AH=01H)のファンクション実行時、現在の文字高さの値によりカーソル形状の計算を行い設定します。

(BL) = 36H ビデオ・スクリーンのON/OFF

＜設定＞
(AL) = 00H スクリーンのON

(AL) = 01H　スクリーンのOFF

＜結果＞

(AL) = 12H　この機能がサポートされている

注:　画面表示のオン／オフを行います。ただしオフの状態でもビデオのI/Oポート、VRAMへアクセスすることができます。

AH=13H　文字列の読み取りと書き込み

文字列の書き込み

＜設定＞

(ES:BP)　ビデオ・バッファーに書き込む文字列が置かれる記憶域の先頭アドレスを指定します

(CX)　書き込まれる文字数（属性バイトは含まない。全角１文字は2に数える）

(DX)　文字列が書き込まれる位置（行番号、桁番号）

(BH)　ページ番号。0に設定してください。

(AL) = 00H　BLに指定された属性に従って、(ES:BP)に指定した記憶域の文字列（文字コード、文字コード、文字コード...）が書き込まれます。このとき、カーソルは移動しません。

(BL)　文字の属性。

(AL) = 01H　記憶域の文字列が書き込まれると同時にカーソルが移動することを除いて**AL=00H**と同じです

(BL)　文字の属性。

(AL) = 02H　(ES:BP)に指定した記憶域の文字列（文字コード、属性、文字コード、属性...）が書き込まれます。このときカーソルは移動しません。文字モードの場合のみ有効です。

(AL) = 03H　記憶域の文字列（文字コード、属性、文字コード、属性...）が書き込まれると同時にカーソルも移動します。文字モードの場合のみ有効です。

注:

1. 文字の属性については、『ビデオ・モードについての補足説明』を参照してください。
2. 復帰、改行、後退、およびベルの各文字は、制御コードとして取り扱われ、画面に表示されません。
3. 画面が1行上方向に移動するときに、最下行に埋められるブランク行の属性は、上方向に移動する前の行の属性によって決まります。

文字ブロックの読み取り

＜設定＞

(ES:BP)　ビデオ・バッファーから読み取られるデータを置く記憶域の先頭アドレスを設定します。

(CX)　読み取られる文字数（属性バイトは含まない）。

(DX)　文字列が読み取られる位置（行番号、桁番号）。

(BH) ページ番号。0に設定してください。

(AL) = 10H 文字列（文字コード、属性、文字コード、属性... の順)が記憶域(ES:BP)に読み取られます。このとき、カーソルは移動しません。

(AL) = 11H 文字列（文字コード、属性0, 属性1, 属性2...の順）が記憶域(ES:BP)に読み取られます。このとき、カーソルは移動しません。ビデオ・モード73H（拡張CGAテキスト・モード）の場合のみ有効です。

注:

1. 画面の表示範囲を越える文字数の読み取りを行うと、読み取り命令は表示範囲を越えたところで停止します。

2. 文字の属性については、『ビデオ・モードについての補足説明』を参照してください。

文字ブロックの書き込み

＜設定＞

(ES:BP) ビデオ・バッファーに書き込まれるデータの記憶域の先頭アドレスを指定します。

(CX) 書き込まれる文字数（属性バイトは含まない。全角１文字は2に数える）

(DX) 文字列が書き込まれる位置（行番号、桁番号）

(BH) ページ番号。0に設定してください。

(AL) = 20H (ES:BP)に指定した記憶域の文字列（文字コード、属性、文字コード、属性...)がビデオ・バッファーに書き込まれます。このとき、カーソルは移動しません。

(AL) = 21H (ES:BP)に指定した記憶域の文字列（文字コード、属性0, 属性1, 属性2...)がビデオ・バッファーに書き込まれます。このとき、カーソルは移動しません。ビデオ・モード73H（拡張CGAテキスト・モード）の場合のみ有効です。

注:

1. 文字の属性については、『ビデオ・モードについての補足説明』を参照してください。

2. 復帰文字、改行文字、バックスペース、および、ベルの各文字は、コマンドとして取り扱われません。

3. 画面の表示範囲を越えて文字を書き込むと、表示範囲を越える文字は廃棄され、画面の上方向の移動も行われません。

AH=14H~17H 予約済み

AH=18H 文字フォント・パターンの要求

＜設定＞

(AL) 文字パターンの読み取りまたは書き込みを行います。

= 00H システムのフォント・バッファーからの文字パターンの読み取り

= 01H システムのフォント・バッファーへの文字パターンの書き込み（ユーザー定義文字のみ）

(BH) 予約済み（0に設定してください）

(BL) フォントの選択

= 00H 文字セット0

= 01H~FFH 予約済み

(CH) 全角文字の場合、内部コード（シフトJIS）の第1バイト目。半角の場合、00H。

(CL) 全角文字の場合、内部コード（シフトJIS）の第2バイト目。半角の場合、文字コード。

(DH,DL) 文字の幅、文字の高さ

08H,10H 8×16ドット半角文字パターン

08H,13H 8×19ドット半角文字パターン

10H,10H 16×16ドット全角文字パターン

18H,18H 24×24ドット全角文字パターン

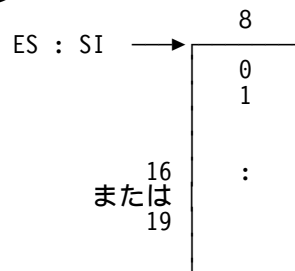
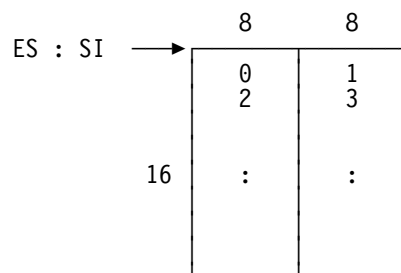
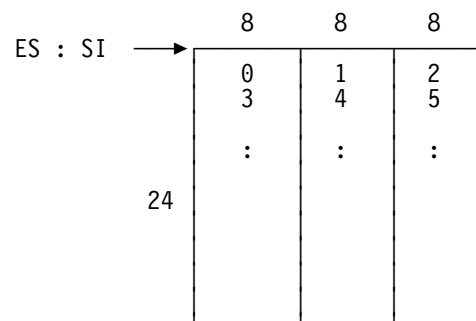
(ES:SI) 文字パターンのイメージ・バッファー（記憶域）の先頭アドレス

＜結果＞

(AL) 戻りコード

= 00H 正常終了

= 01H~FFH エラー

文字パターン

8 x 16 (1バイト文字セット)および 8 x 19

16 x 16 (2バイト文字セット)　　24 x 24 (2バイト文字セット)

注:

1. 日本語モードにおけるBIOSは、文字セット0のみサポートします。BIOSは、文字の書き込みインターフェースにおいてフォントの選択を許可しません。
2. アプリケーション・プログラムは、このBIOS割り込みを用いて、システムで使用可能なすべての文字パターンを読み取ることができます。
3. この機能は、内部的にINT15H, AH = 50H（フォントの読み取りと書き込み）の機能を呼び出します。高速な処理を必要とする場合はINT15H, AH = 50Hの機能を利用してください。

AH=19H 予約済み

AH=1AH ディスプレイ組み合わせコードの読み取り

＜設定＞

(AL) = 00H ディスプレイ組み合わせコードを読み取ります。

＜結果＞

(AL) = 1AH 機能がサポートされている

(BH) 予約済み

(BL) 活動状態のディスプレイ・コード

= 00H	ディスプレイなし
= 01H~06H	予約済み
= 07H	VGA（モノクロ）
= 08H	VGA（カラー）
= 09H~FFH	予約済み

(AL) = その他の値 予約済み

AH=1BH~1CH 予約済み

AH=1DH キーボード・シフト標識域の表示と消去

＜設定＞

(AL)

= 00H	キーボード・シフト標識域を表示します。
= 01H	キーボード・シフト標識域を消去します。
= 02H	キーボード・シフト標識域の状況を読み取ります。
= 他の値	予約済み

(BX) 標識域の行数（AL = 00Hの場合のみ）

＜結果＞ AL = 02Hの場合のみ返されます。

(BX) 標識域の行数。

注:

1. この機能は入力支援サブシステム($IAS.SYS)に予約済みです。アプリケーション・プログラムは、この機能を呼びだしてキーボード・シフト標識域の表示や消去を行ってはいけません。
   シフト標識域の表示／非表示をアプリケーション・プログラムが行う場合はINT16h (AH = 14h)を使用してください。

2. この機能は、画面下部のキーボード・シフト標識域を表示または消去するときのみ使用されます。画面下部が標識域行に予約されると、‘画面の行数–1(BIOSデータ域0040:0084（バイト）)’が(BX)に指定された行数分減ります。

3. 『AH=00H　ビデオ・モードの設定』において、キーボード・シフト標識域は、自動的に消去されますが、入力支援サブシステムが導入されている場合は、この機能により、ビデオ・モードの設定前と設定後もシフト標識域はそのまま維持されます。

AH=1EH~FDH 予約済み

AH=FEH ビデオ・バッファー・アドレスの読み取り

＜設定＞

(ES:DI) ビデオ・バッファーのアドレスを指定します。
= B800:0000H（ビデオ・モード03Hの場合）

＜結果＞

(ES:DI) ビデオ・バッファーの実際のアドレス

ハードウェア・ビデオ・バッファーが存在する場合、このアドレスは不変です。存在しない場合、擬似ビデオ・バッファーのアドレスに変ります。

注: この機能はビデオ・モード03Hで使用可能です。

AH=FFH 画面表示の更新

＜設定＞

(ES:DI) 更新されるビデオ・バッファー内の先頭アドレスを設定します。

(CX) ビデオ・バッファーに書き込まれる文字数を指定します。

注:

1. この機能は、ビデオ・モード03Hで使用可能です。
2. ハードウェア・ビデオ・バッファーが存在する場合、この機能は効力がありません。

INT 11H 装置構成情報

装置の接続状況が得られます。AXの各ビットは次のような意味を持ちます。

<結果>

ビット15,14	プリンター数
ビット13	予約済み
ビット12	予約済み
ビット11〜9	ASYNC通信ポート数
ビット8	予約済み
ビット7, 6	フロッピーディスク数
=00	1ドライブ
=01	2ドライブ
ビット5, 4	予約済み
ビット3	未使用
ビット2	マウスを接続のとき 1
ビット1	数値演算プロセッサーを装着のとき 1
ビット0=1	予約済み（常に1が返されます）

INT 12H 記憶域サイズを得る

AXに1Kバイト単位の連続した記憶域サイズが得られます。

INT 13H フロッピーディスク入出力

AH=00H フロッピーディスク・システムのリセット

フロッピーディスク・ドライブがリセットされます。その結果はCFとAHに返されます。

<設定>

(DL) ドライブ番号 (0, 1, ...)
ビット7 = 0 フロッピーディスク・ドライブを示します。

<結果>

CF キャリー・フラグにフロッピーディスク操作状況が返されます。
= 0 フロッピーディスクの操作状況が00H
= 1 フロッピーディスクの操作状況が00H以外

(AH) フロッピーディスクの操作状況が返されます。状況については、
『AH=01H フロッピーディスク状況の読み取り』を参照してください。

注:

1. フロッピーディスク入出力ルーチンよりエラーが返された場合、フロッピーディスク・システムのリセットを実行してから、命令をやり直してください。
2. (DL)に80H以上の値を指定した場合、フロッピーディスク・システムがリセットされ、ディスク・システムもリセットされます。命令が完了した後に、'AH=01H フロッピーディスク状況の読み取り'を実行してフロッピーディスク・システムの状況を調べてください。

AH=01H フロッピーディスク状況の読み取り

最後にフロッピーディスクに対して実行された命令の状況を読み取ります。

<設定>

(DL) ドライブ番号 (0, 1, ...)
ビット7 = 0 フロッピーディスク・ドライブ（値はチェックされます）

<結果>

CF キャリー・フラグにフロッピーディスク操作状況が返されます。
= 0 フロッピーディスクの操作状況が00H
= 1 フロッピーディスクの操作状況が00H以外

(AH) フロッピーディスクの操作状況
= 80H フロッピーディスク・ドライブ作動不可
= 40H SEEK 操作の障害
= 20H 制御装置の障害
= 10H フロッピーディスクの読み取りにおけるCRC (Cyclic Redundancy Check)エラー
= 0CH 指定されたメディア・タイプが見つからない
= 09H 64KB 境界にまたがるDMA (Direct Memory Access)へのアクセス

= 08H　操作におけるDMAオーバーラン
= 06H　フロッピーディスクの入れ替えが行われた
= 04H　要求されたセクター番号が見つからない
= 03H　書き込み禁止フロッピーディスクへの書き込み
= 02H　アドレス・マークが見つからない
= 01H　無効なフロッピーディスク・パラメーターが渡された
= 00H　エラーなし、正常終了

AH=02H　セクターの読み取り

フロッピーディスクからデータを読み取ります。読み取られたデータは、(ES:BX)に指定した記憶域に記憶されます。

＜設定＞

(DL)　ドライブ番号 (0, 1, ...)
ビット7 = 0　フロッピーディスク・ドライブ（値はチェックされます）

(DH)　ヘッド番号（0, 1。値はチェックされません）
(CH)　トラック番号 (0, 1, ...)
(CL)　セクター番号（値はチェックされません）
(AL)　セクター数（値はチェックされません）
(ES:BX) バッファー・アドレス

＜結果＞

CF　キャリー・フラグにフロッピーディスクの操作状況が返されます。
= 0　フロッピーディスクの操作状況が00H
= 1　フロッピーディスクの操作状況が00H以外
(AL)　実際にバッファーに送られたセクター数
(AH)　命令により返されたフロッピーディスクの操作状況（状況については、『AH=01H　フロッピーディスク状況の読み取り』を参照してください。）

注:　フロッピーディスク入出力ルーチンよりエラーが返された場合、フロッピーディスク・システムをリセットした後に、命令を実行し直してください。

AH=03H　セクターへの書き込み

フロッピーディスクにデータを書き込みます。書き込むデータは、(ES:BX)に指定された記憶域に準備しておきます。

＜設定＞

(DL)　ドライブ番号 (0, 1, ...)
ビット7 = 0　フロッピーディスク・ドライブ（値はチェックされます）

(DH)　ヘッド番号（0, 1。値はチェックされません）
(CH)　トラック番号 (0, 1, ...)
(CL)　セクター番号（値はチェックされません）
(AL)　セクター数（値はチェックされません）
(ES:BX) バッファー・アドレス

＜結果＞

CF　キャリー・フラグにフロッピーディスクの操作状況が返されます。

= 0　　フロッピーディスクの操作状況が00H
= 1　　フロッピーディスクの操作状況が00H以外

(AL)　実際にバッファーに送られたセクター数

(AH)　命令により返されたフロッピーディスクの操作状況（状況については、『AH=01H　フロッピーディスク状況の読み取り』を参照してください。）

注:　フロッピーディスク入出力ルーチンよりエラーが返された場合、フロッピーディスク・システムをリセットした後に、命令を実行し直してください。

AH=04H　セクターの検査

フロッピーディスクのデータを読み取り検査します。

＜設定＞

(DL)　ドライブ番号 (0, 1, ...)
　ビット7 = 0　　フロッピーディスク・ドライブ（値はチェックされます）

(DH)　ヘッド番号 （0,1。値はチェックされません）

(CH)　トラック番号 (0, 1, ...)

(CL)　セクター番号 （値はチェックされません）

(AL)　セクター数 （値はチェックされません）

(ES:BX) バッファー・アドレス

＜結果＞

CF　キャリー・フラグにフロッピーディスクの操作状況が返されます。
= 0　　フロッピーディスクの操作状況が00H
= 1　　フロッピーディスクの操作状況が00H以外

(AL)　実際にバッファーに送られたセクター数

(AH)　命令により返されたフロッピーディスクの操作状況（状況については、『AH=01H　フロッピーディスク状況の読み取り』を参照してください。）

注:　フロッピーディスク入出力ルーチンよりエラーが返された場合、フロッピーディスク・システムをリセットした後に、命令を実行し直してください。

AH=05H　トラックのフォーマット

フロッピーディスク上の（一部）アドレス・フィールドをフォーマットします。バッファー・ポインター(ES:BX)は、トラックのアドレス・フィールドを指定します。アドレス・フィールドの持つ4バイトの内容を以下に示します。

バイト 0　　トラック番号
バイト 1　　ヘッド番号
バイト 2　　セクター番号
バイト 3　　セクター当りのデータ・バイト数
= 00H　　128バイト/セクター
= 01H　　256バイト/セクター
= 02H　　512バイト/セクター
= 03H　　1024バイト/セクター

トラック上のセクター毎にアドレス・フィールドがなければなりません。このアドレス・フィールドの情報は、セクターの読み取りと書き込みを実行中に要求したセクターを見つけるために使用されます。ドライブが複数のフォーマットをサポートしている場合、フロッピーディスクをフォーマットする前に、フォーマットされるフロッピーディスク・タイプを設定するために、『AH=18H　メディア・タイプの設定』を実行してください。

AH=06H~07H　予約済み

AH=08H　フロッピーディスクの情報の読み取り

＜設定＞

(DL) ドライブ番号 (0, 1, ...)
ビット7 = 0　フロッピーディスク・ドライブ（値はチェックされます）

＜結果＞

(ES:DI) 情報を得るドライブがサポートしている最大のメディア・タイプの11バイトのパラメーター・テーブルを指し示します（『フロッピーディスク・ドライブのパラメーター・テーブル』を参照）。

(CH) 最大トラック数（トラック数10ビット中の低位8ビット；0 ~）

(CL)
ビット7, 6　最大トラック数（トラック数10ビット中の高位2ビット；0~)
ビット5~0　トラック当りの最大セクター数

(DH) 最大ヘッド番号

(DL) 取り付けられているフロッピーディスク・ドライブの数

(BH) 常に0

(BL)
ビット7~4　常に0
ビット3~0　フロッピーディスク・ドライブのタイプ(CMOS内)
= 02H　1.2MB, 5.25インチ、80トラック
= 04H　1.44MB, 3.5インチ、80トラック
= 06H　予約済み

*MB = 1,048,576バイト

(AX) 常に0

システムのバッテリー切れ、またはCMOSの異常以外でドライブ・タイプを知ることができる場合は、BLに00Hが返され、他のすべてのレジスターに上記の値が返されます。

要求したフロッピーディスク・ドライブが取り付けられていなければ、(AX), (BX), (CX), (DX), (DI), および(ES)に、00Hが返されます。

フロッピーディスクの操作状況を示すBIOSデータ域の16進 40:41 = 0 および CF = 0

AH=9H~14H 予約済み

AH=15H フロッピーディスク・ドライブのタイプの読み取り

＜設定＞

(DL) ドライブ番号 (0, 1, ...)

ビット7 = 0 フロッピーディスク・ドライブ（値はチェックされます）

＜結果＞

CF = 0 命令は正常に完了

(AH) = 00H ドライブが取り付けられていない

= 01H フロッピーディスクの入れ替えを検知できない

= 02H フロッピーディスクの入れ替えを検知できる

= 03H 予約済み

BIOSデータ域 16進 40:41 にフロッピーディスクの操作状況

AH=16H フロッピーディスク入れ替え状況

＜設定＞

(DL) ドライブ番号 (0, 1, ...)

ビット7 = 0 フロッピーディスク・ドライブ（値はチェックされます）

＜結果＞

(AH) = 00H ‘フロッピーディスク入れ替え’信号を検知していない

= 01H 無効なフロッピーディスク・パラメーター

= 06H ‘フロッピーディスク入れ替え’信号を検知した

= 80H フロッピーディスク・ドライブ作動不可

CF = 0 フロッピーディスクの操作状況が00H

= 1 フロッピーディスクの操作状況が00H以外

AH=17H 予約済み

AH=18H　メディア・タイプの設定

‘AH=05H トラックのフォーマット’を呼び出す前に、この機能を呼び出します。フロッピーディスクが入れ替えられた場合にもこの機能を呼び出す必要があります。

この機能が正常終了するには、ドライブにフロッピーディスクが入っていなければなりません。

＜設定＞

(DL)　ドライブ番号 (0, 1, ...)
　ビット7 = 0　フロッピーディスク・ドライブ（値はチェックされます）

(CH)　フォーマットするメディアのトラック数（0から始まる、トラック数10ビット中の低位8ビット）

(CL)
　ビット7, 6　フォーマットするメディアのトラック数（0から始まる、トラック数10ビット中の高位2ビット）
　ビット5~0　セクター数/トラック（1から始まる）

＜結果＞

CF = 0　フロッピーディスクの操作状況が00H
　= 1　フロッピーディスクの操作状況が00H以外

(ES:DI)　このメディア・タイプに関する11バイトのパラメーター・テーブルへのポインター。AHが0以外の場合、不変（『フロッピーディスク・ドライブのパラメーター・テーブル』を参照）

(AH)　フロッピーディスクの操作状況（『AH=01H　フロッピーディスク状況の読み取り』の値を参照)

注:　この機能は‘フロッピーディスクの入れ替え’信号を監視しています。この信号が検知されると、‘フロッピーディスクの入れ替え’信号をオフ状態にするように回路が働きます。この試みが成功すれば（たとえば、フロッピーディスクがドライブに差し込まれていないとき）、BIOSはAHに80H（フロッピーディスク作動不可）とCFに1を返します。

‘フロッピーディスクの入れ替え’信号がオフ状態のときに、BIOSは要求のあった機能を実行します。

AH=19H~FFH　予約済み

INT　13H　ハード・ディスク入出力

注:

1. 予約済みのすべての入力フィールドは、常に0に設定してください。
2. この節で説明するハードディスク入出力ルーチンの機能においてドライブ番号をDLに指定する必要のある機能に関しては、DLの値がチェックされます。
3. ドライブ番号のビット7は、ハードディスクのBIOSルーチンを実行するのに1に設定しておく必要があります。

AH=00H　ディスク・システムのリセット

ディスク・ドライブがリセットされます。結果はCFとAHに返されます。

＜設定＞

(DL)　ドライブ番号 (0, 1, ...)
　ビット 7 = 1　ハードディスク・ドライブ（値はチェックされます）

＜結果＞

CF = 1　ディスクの操作状況が0以外
= 0　ディスクの操作状況が0
(AH)　ディスクの操作状況（『AH=01H　ハード・ディスク状況の読み取り』を参照）

注:

1. ディスク・システムのリセットは、ドライブ番号（ビット7を除く）がハードディスク・ドライブの最大数以下の場合のみ、実行されます。フロッピーディスク・システムは、DLのすべての値に関してリセットされます。

AH=01H　ハード・ディスク状況の読み取り

最後にディスクに対して実行された命令の状況を読み取ります。

＜設定＞

(DL)　ドライブ番号 (0, 1, ...)
　ビット7 = 1　ハードディスク・ドライブ

＜結果＞

CF = 1　ディスクの操作状況が0以外
= 0　ディスクの操作状況が0

(AH)　ディスクの操作状況
　= 00H　エラーなし、正常終了
　= 01H　無効な機能またはパラメーターを指定
　= 02H　アドレス・マークが見つからない
　= 04H　セクターが見つからない
　= 05H　リセットに失敗
　= 07H　ドライブ・パラメーター・アクティビティーが失敗

= 08H　操作におけるDMAオーバラン
= 09H　データ境界エラー
= 0AH　不良セクター・フラグ検出
= 0BH　不良シリンダー検出
= 0DH　フォーマットにおける無効なセクター数
= 0EH　制御データ・アドレス・マーク検出
= 0FH　DMAのアービトレーション・レベルが範囲外
= 10H　ECC (Error Checking and Correction)の訂正不可またはCRC (Cyclic Redundancy Check)エラー
= 11H　ECC訂正データ・エラー
= 12H　コマンド処理中
= 13H　ドライブの電源が入っていない
= 20H　制御装置の障害
= 40H　SEEK操作の障害
= 80H　タイム・アウト
= AAH　ドライブ作動不可
= BBH　確定できないエラー発生
= CCH　選択したドライブにおける書き込み障害
= E0H　状況エラー／エラー・レジスター = 0
= FFH　Sense操作の失敗

注:　ディスク・システムのリセットは、ドライブ番号（ビット7を除く）がハードディスク・ドライブの最大数以下の場合のみ、実行されます。フロッピーディスク・システムは、DLのすべての値に関してリセットされます。

AH=02H　セクターの読み取り

ハードディスクからデータを読み取ります。読み取られたデータは、(ES:BX)に指定した記憶域に記憶されます。

＜設定＞

(DL)　ドライブ番号 (0, 1, ...)
　ビット 7 = 1　ハードディスク・ドライブ
(DH)　ヘッド番号（0から始まる。値はチェックされません）
(CH)　シリンダー番号（シリンダー番号10ビット中の低位8ビットを指定、0から始まる。値はチェックされません。）
(CL)
　ビット7, 6　シリンダー番号（シリンダー番号10ビット中の高位2ビット、0から始まる。値はチェックされません）
　ビット5~0　セクター番号（値はチェックされません）
(AL)　セクター数
(ES:BX) バッファー・アドレス

＜結果＞

CF = 1　ディスクの操作状況が0以外
　= 0　ディスクの操作状況が0
(AH)　ディスクの操作状況（『AH=01H　ハード・ディスク状況の読み取り』を参照）

注:

1. AH=11Hのエラーは、読み取られたデータがECCアルゴリズムにより訂正された回復可能なエラーであることを示します。データは正しいと考えられます。ただし、BIOSルーチンは、制御プログラムにこの判別を行わせるためにエラーとして示します。エラーは、データが上書きされれば、再発することはありません。

2. ディスク入出力ルーチンによりエラーが報告された場合、ディスク・システムをリセットしてから、命令をやり直してください。

AH=03H セクターへの書き込み

フロッピーディスクにデータを書き込みます。書き込むデータは、(ES:BX)に指定された記憶域に準備します。

＜設定＞

(DL) ドライブ番号 (0, 1, ...)
ビット 7 = 1 ハードディスク・ドライブ

(DH) ヘッド番号（0から始まる。値はチェックされません）

(CH) シリンダー番号（シリンダー番号10ビット中の低位8ビットを指定、0から始まる。値はチェックされません。）

(CL)
ビット7, 6 シリンダー番号（シリンダー番号10ビット中の高位2ビット、0から始まる。値はチェックされません）
ビット5~0 セクター番号（値はチェックされません）

(AL) セクター数

(ES:BX) バッファー・アドレス

＜結果＞

CF = 1 ディスクの操作状況が0以外
= 0 ディスクの操作状況が0

(AH) ディスクの操作状況（『AH=01H　ハード・ディスク状況の読み取り』を参照）

注: エラーがハードディスクBIOSにより返された場合、ディスク・システムをリセットしてから、命令をやり直してください。

AH=04H セクターの検査

ディスクのデータを読み取り検査します。

＜設定＞

(DL) ドライブ番号 (0, 1, ...)
ビット 7 = 1 ハードディスク・ドライブ

(DH) ヘッド番号（0から始まる。値はチェックされません）

(CH) シリンダー番号（シリンダー番号10ビット中の低位8ビットを指定、0から始まる。値はチェックされません。）

(CL)
ビット7, 6 シリンダー番号（シリンダー番号10ビット中の高位2ビット、0から始まる。値はチェックされません）
ビット5~0 セクター番号（値はチェックされません）

(AL) セクター数
(ES:BX) バッファー・アドレス

＜結果＞
CF = 1 ディスクの操作状況が0以外
= 0 ディスクの操作状況が0
(AH) ディスクの操作状況（『AH=01H　ハード・ディスク状況の読み取り』を参照）

注: エラーがハードディスクBIOSにより返された場合、ディスク・システムをリセットしてから、命令をやり直してください。

AH=05H シリンダーのフォーマット

ディスク上の（一部）アドレス・フィールドをフォーマットします。

＜設定＞
(DL) ドライブ番号 (0, 1, ...)
ビット 7 = 1　ハードディスク・ドライブ
(DH) ヘッド番号（0から始まる。値はチェックされません）
(CH) シリンダー番号 (シリンダー番号10ビット中の低位8ビットを指定、0から始まる。値はチェックされません)
(CL)
ビット7, 6　シリンダー番号（10ビット内の高位2ビット、0から始まる。値はチェックされません）
(ES:BX) バッファー・アドレス
512バイト・バッファーを指定します。最初の2×（シリンダー当たりのセクター数）バイトには、セクター毎のFとNが含まれます。
F = 00H 良セクター
= 80H 不良セクター
N セクター番号

＜結果＞
CF = 1 ディスクの操作状況が0以外
= 0 ディスクの操作状況が0
(AH) ディスクの操作状況（『AH=01H　ハード・ディスク状況の読み取り』を参照）

注: エラーがハードディスクBIOSにより返された場合、ディスク・システムをリセットしてから、命令をやり直してください。

AH=08H ハード・ディスク情報の読み取り

＜設定＞
(DL) ドライブ番号 (0, 1, ...)
ビット 7 = 1　ハードディスク・ドライブ

＜結果＞
(DL) 連結されているドライブの数（1,2;コントローラー・カード0の合計のみ）
(DH) ヘッド番号の最大数（範囲 0~3FH）

(CH)　シリンダー番号の最大数の低位8ビット
(CL)　セクターの最大数とシリンダー番号の最大数の高位2ビット

ドライブ番号が無効な場合、AHとBIOSデータ域16進 40:74（最後のディスクの操作状況）=07H, CXとDX=0, およびCF=1が設定されます。ハードディスクが接続されていないか、ハードディスクのアダプター・カードが取り付けられていなければ、AHとBIOSデータ域 16進 40:41（最後のディスクの操作状況）=01H, およびCF=1が設定されます。

AH=09H ドライブの初期化

割り込み41Hはドライブ0に関する1つのパラメーター・テーブルを指し示し、割り込み46Hはドライブ1に関する1つのパラメーター・テーブルを指し示します。DLに80Hが設定された場合、割り込み41Hを使ってドライブ0が初期設定されます。DLに81Hが設定された場合、割り込み46Hを使ってドライブ1が初期設定されます。DLに他の値を設定した場合、無効なコマンド状況が返されます。

＜設定＞
(DL)　ドライブ番号 (0, 1, ...)
　ビット7 = 1　ハードディスク・ドライブ

＜結果＞
CF = 1　ディスクの操作状況が0以外
= 0　ディスクの操作状況が0
(AH)　ディスクの操作状況（『AH=01H　ハード・ディスク状況の読み取り』を参照）

AH=0AH~0BH 予約済み

AH=0CH SEEK

ハードディスク上のデータの読み・書きを行うために、目的のシリンダーにヘッドを移動します。

＜設定＞
(DL)　ドライブ番号 (0, 1, ...)
　ビット7 = 1　ハードディスク・ドライブ
(DH)　ヘッド番号（0から始まる。値はチェックされません）
(CH)　シリンダー番号（シリンダー番号10ビット中の低位8ビット、0 から始まる。値はチェックされません。）
(CL)
　ビット7, 6　シリンダー番号（シリンダー番号の10ビット中の高位2ビット、0から始まる。値はチェックされません。）

＜結果＞
CF = 1　ディスクの操作状況が0以外
= 0　ディスクの操作状況が0
(AH)　ディスクの操作状況（『AH=01H　ハード・ディスク状況の読み取り』を参照）

注: エラーがハードディスクBIOSにより返された場合、ディスク・システムをリセットしてから、命令をやり直してください。

AH=0DH 代替ディスク・リセット

＜設定＞

(DL) ドライブ番号 (0, 1, ...)
ビット7 = 1 ハードディスク・ドライブ

＜結果＞

CF = 1 ディスクの操作状況が0以外
= 0 ディスクの操作状況が0
(AH) ディスクの操作状況（『AH=01H ハード・ディスク状況の読み取り』を参照）

注: 代替ディスク・リセットは、ビット7を除くドライブ番号がハード・ディスク・ドライブの最大数以下であれば、実行されます。

AH=0EH~10H 予約済み

AH=11H ヘッドの位置合わせ

ヘッドの移動により、ヘッドとシリンダー／セクター番号との位置にズレが生じた場合、この機能を実行して、ヘッドを0シリンダーの位置に戻して、ズレを解消します。

＜設定＞

(DL) ドライブ番号 (0, 1, ...)
ビット7 = 1 ハードディスク・ドライブ

＜結果＞

CF = 1 ディスクの操作状況が0以外
= 0 ディスクの操作状況が0
(AH) ディスクの操作状況（『AH=01H ハード・ディスク状況の読み取り』を参照）

注: ディスク入出力ルーチンよりエラーが返された場合、ディスク・システムをリセットし、命令をやり直してください。

AH=12H~14H 予約済み

AH=15H ハードディスク・ドライブのタイプの読み取り

＜設定＞

(DL) ドライブ番号 (0, 1, ...)
ビット7 = 1 ハードディスク・ドライブ

＜結果＞

CF = 0 命令は正常に完了
(AH) = 00H ドライブが接続されていないか、DLが無効
= 01H 予約済み

= 02H 予約済み
= 03H ハードディスク

(CX,DX) 512バイト・ブロックの数
AH = 00Hの場合、CXとDX = 0です。
(DX) 低位ワード
(CX) 高位ワード

AH=16H~19H 予約済み

ROMテーブル

BIOSにより使用されるROMテーブルを以下に示します。ROMテーブルにはシステムまたはアダプターBIOSによりサポートされるハードウェア装置の特性を定義しています。

ハードディスク・ドライブのパラメーター・テーブル

ハードディスク・ドライブのパラメーター・テーブルを以下に示します。

オフセット	データ長	説明
0	1ワード	最大シリンダー数
2	1バイト	最大ヘッド数
3	1ワード	未使用
5	1ワイド	代替シリンダー書き込み開始
7	1バイト	未使用
8	1バイト	制御バイト
		ビット7- 再試行不可 または
		ビット6 - 再試行不可
		ビット5 - 工場出荷時の不良 シリンダーのマップ表示
		ビット3 -ヘッドが8以上
		ビット2, 1, 0 -予約済み
9	1バイト	未使用
10	1バイト	未使用
11	1バイト	未使用
12	1ワード	ランディング領域
14	1バイト	トラック当りのセクター数
15	1バイト	予約済み

図 3. ハードディスク・ドライブのパラメーター・テーブルの定義

注: ハードディスク・ドライブのパラメーター・テーブルは、ESDIハードディスクまたはSCSIハードディスクには使用されません。

フロッピーディスク・ドライブのパラメーター・テーブル

フロッピーディスク・ドライブのパラメーター・テーブルは次のとおりです。

オフセット	データ長	説明
0	1バイト	1番目の指定バイト
1	1バイト	2番目の指定バイト
2	1バイト	フロッピーディスク・ドライブの回転を停止するまでの待ち時間(55ミリ秒)
3	1バイト	セクター当りのバイト数 00H = 128バイト／セクター 01H = 256バイト／セクター 02H = 512バイト／セクター 03H = 1024バイト／セクター
4	1バイト	トラック当りのセクター数
5	1バイト	ギャップの長さ
6	1バイト	データの長さ
7	1バイト	フォーマットに関するギャップの長さ
8	1バイト	フォーマットに関するバイトの数
9	1バイト	ヘッドの安定時間（ミリ秒）
10	1バイト	モーター起動時間（1/8秒） たとえば、8は1秒を示します。

図 4. フロッピーディスク・ドライブのパラメーター・テーブル

注: フロッピーディスク・ドライブ・パラメーター・テーブルは、INT 1EHにより指し示されます。

INT 14H ASYNC入出力

- シリアルXは物理ASYNCポートを指定します。

  シリアル１ー Ｉ／Ｏポート 03F8H～03FFH, 割り込みレベル４
  シリアル２ー 〃 02F8H～02FFH, 〃 ３

  物理ASYNCポートは、ASYNC通信BIOSを用いずにH/Wを直接制御する場合に使用します。

- COMxは論理ASYNCポートを指定します。

  COM1 ー最初に見つかった物理ASYNCポート(シリアル1, 2のいずれか)
  COM2 ー２番目に 〃

  (DX)=0, 1で論理ASYNCポートCOM1, COM2を指定します。

論理ASYNCポートCOM1, COM2に対応する物理ASYNCポートのアドレスは、それぞれBIOSデータ域40:00, 40:02に格納されています。

ASYNC通信BIOSでは、RS232C通信ポートのバイト・ストリーム入出力機能を以下のパラメーターに従って提供します。(DX)には論理通信ポート番号(0, 1)を指定します。

AH=00H 通信ポートの初期設定

＜設定＞

(AL) 初期設定する値を指定します。

ビット7, 6, 5	ボー・レート	
	= 000	110 bps
	= 001	150bps
	= 010	300bps
	= 011	600bps
	= 100	1200bps
	= 101	2400bps
	= 110	4800bps
	= 111	9600bps
ビット4, 3	パリティー	
	= x0	なし
	= 01	奇数
	= 11	偶数
ビット2	ストップ・ビット	
	= 0	1ストップ・ビット
	= 1	2ストップ・ビット
ビット1, 0	ワード長	
	= 10	7ビット
	= 11	8ビット

(DX)　論理ASYNCポート(0, 1)を指定します。

＜結果＞

(AL)　モデム状況（『AH=03H　通信ポート状況』を参照）

(AH)　回線制御状況（『AH=03H　通信ポート状況』を参照）

注:　回線制御状況のビット7 = 1 の場合、他のビットが表す状況は不確実です。

AH=01H　文字の送信

ALに指定した文字が通信回線に送信されます。このとき、DTR信号およびRTS信号がオンにされます。

回線にデータ・バイトを送信できなかったとき（規定時間内にDSR信号およびCTS信号がオンにならなかったとき）は、AHのビット7が1に設定されて戻されます。AHの残りのビットは、通信ポート状況(AH=03H)の場合と同様、現在の回線の状況を返します。

＜設定＞

(AL)　送信する文字を指定します。

(DX)　論理ASYNCポート(0, 1)を指定します。

＜結果＞

(AL)　データは保持されています。

(AH)　回線制御状況（『AH=03H　通信ポート状況』を参照）

AH=02H　文字の受信

回線から受信した１文字をALに返します。通信ポート状況(AH=03H)の場合と同様、AHに現在の回線状況が返されます。ただし、このルーチンの場合オンにされるのはエラー・ビット(7, 4, 3, 2, 1)だけです。タイム・アウト・ビットは規定時間内にDSR信号がオンにならなかったことを示します。DTR信号はオン、RTS信号はオフにされます。

＜設定＞

(DX)　論理ASYNCポート(0, 1)を指定します。

＜結果＞

(AL)　受信された文字

(AH)　回線制御状況（『AH=03H　通信ポート状況』を参照）

注:　文字を受信するまで、またはタイム・アウトになるまでルーチンは待ち状態になります。

AH=03H 通信ポート状況

通信ポート状況をAXに返します。

AH(回線制御状況)

ビット7	タイム・アウト
ビット6	送信用シフト・レジスターが空
ビット5	送信用保持レジスターが空
ビット4	ブレークを検出
ビット3	フレーム・エラー
ビット2	パリティー・エラー
ビット1	オーバーラン・エラー
ビット0	データ・レディー

AL(モデム状況)

ビット7	受信回線信号の検出
ビット6	呼び出し信号(RI)受信
ビット5	データ・セット・レディー(DSR)
ビット4	送信可(CTS)
ビット3	受信回線信号の検出に変化あり
ビット2	呼び出し信号(RI)受信の終端検出
ビット1	データ・セット・レディーに変化あり
ビット0	送信可に変化あり

AH=04H~FFH 予約済み

INT 15H システム・サービス入出力

AH=49H BIOSタイプの取得

このルーチンを利用して現在のBIOSタイプを知ることができます。

<設定>

AL = 00H BIOSタイプを得る

<結果>

(BL) BIOSタイプ
00H DOS/V BIOS
他の値 予約済み

CY = 1, AH = 86H この機能はサポートされていない
CY = 0, AH = 00H この機能はサポートされている

AH=4FH キーボード・インターセプト

キーボード・インターセプトは、ハードウェア割り込みINT 09H（キーボード）ルーチンにより呼び出され、押されたキーのコードを変更したり、そのまま読み取るために使用されます。通常はシステムは走査コードに変更を加えずに返しますが、オペレーティング・システムは割り込みINT 15Hがアプリケーション・プログラムに渡るのを横取りして以下のどちらか１つを実行します。

1. (AL)を別の走査コードで置き換え、キャリー・フラグを１にセットして返します。押されたキー・コードを変更したことになります。
2. (AL)のキー・コードを処理して、キャリー・フラグを０にリセットします。INT 09Hルーチンは、そのキー・コードを処理しません。

<設定>

(AL) 走査コード

<結果>

(AL) 新しい走査コード
CF = 1

または、

(AL) 走査コードは変更なし
CF = 0

AH=50H フォントの読み取りと書き込み

文字パターンの‘読み取り’と‘書き込み’を行うために、‘フォントの読み取り’と‘フォントの書き込み’機能のアドレスを取得します。各アドレスを読み取るために、次の設定が必要です。

‘フォントの読み取り’機能のアドレスの取得

＜設定＞
AL = 00H
(BH)　フォントの種類
　　ビット0 = 0　1バイト文字セット
　　　　　= 1　2バイト文字セット
　　ビット1~7　予約済み
(BL)　予約済み（0に設定）
(DH,DL) フォント・サイズ（文字幅、文字の高さ）
(BP)　コード・ページ（『コード・ページ』を参照）

＜結果＞
CF　キャリー・フラグ
　　= 0　正常に完了（AH=0の場合）
　　= 1　エラーにより未完了
(AH)　完了コード（『完了コード』を参照）
(ES:BX) 指定した機能のアドレス（AH=0の場合）

‘フォントの書き込み’機能のアドレスの取得

＜設定＞
AL = 01H
(BH)　フォントの種類
　　ビット0 = 0　1バイト文字セット
　　　　　= 1　2バイト文字セット
　　ビット1~7　予約済み
(BL)　予約済み（0に設定）
(DH,DL) フォント・サイズ（文字の幅、文字の高さ）
(BP)　コード・ページ（『コード・ページ』を参照）

＜結果＞
CF　キャリー・フラグ
　　= 0　正常に完了（AH=0の場合）
　　= 1　エラーにより未完了
(AH)　完了コード（『完了コード』を参照）
(ES:BX) 指定した機能のアドレス（AH=0の場合）

コード・ページ

指定できるコード・ページは、以下のとおりです。

0　コード・ページの省略値

完了コード

返されるコードは、以下のとおりです。

00H　正常に完了
01H　無効なフォントの種類が指定された（BHの指定が無効）
02H　BLの指定が無効です（BLは、0に設定しなければなりません）
03H　無効なフォント・サイズが指定された（DXの指定が無効）
04H　無効なコード・ページが指定された（BPの指定が無効）
86H　機能はサポートされていない

文字イメージの読み出しと書き込み

‘フォントの読み取り’と‘フォントの書き込み’を実行するには、次のパラメーターを指定して‘フォントの読み取り’機能の戻りアドレスまたは‘フォントの書き込み’機能の戻りアドレスを呼び出す必要があります。

注: サポートされているフォントのすべてが書き込み可能というわけではありません。完了コードを必ず調べてください。

＜設定＞

(CX)　文字コード
　00xxH　半角文字の場合、CLのみ
　xxxxH　全角文字の場合、内部コードをCHに第1バイト、CLに第2バイト
　(ES:SI)　文字イメージ・バッファーへのポインター

＜結果＞

(AL)　完了コード
　= 0　正常に完了
　= 5　無効なコードが指定された（CXの指定が無効）
　= 6　指定されたフォントは読み取り専用で書き込むことはできない

文字パターンのデータ

読み取りおよび書き込まれる文字パターンのデータは、以下のとおりです。

8 x 16 (1バイト文字セット)および 8 x 19

12 x 24 (1バイト文字セット)

16 x 16 (2バイト文字セット)

24 x 24 (2バイト文字セット)

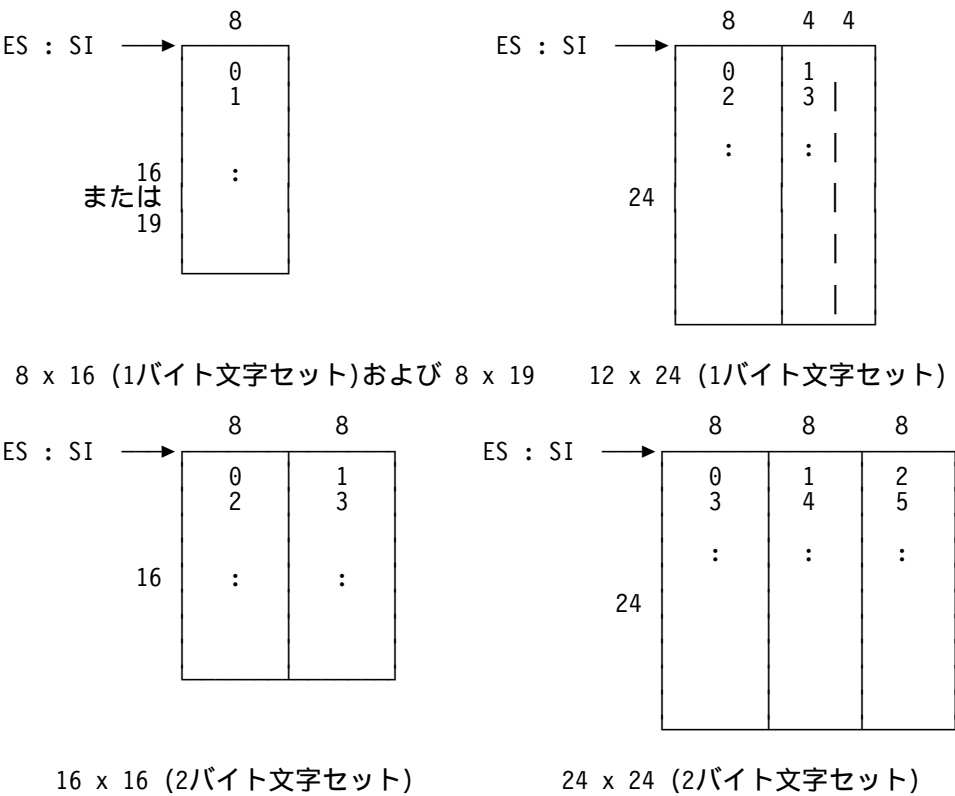



図 5. 文字パターンのデータ

文字パターンの読み取り例

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; 2バイト文字セットの文字パターンを読み取る例;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
read_DBCS_font          dd      ?
DBCS_font_buffer        db      32 dup(?)
;
;       これは、初期化ルーチンです。
;       16x16 2バイト文字セットが使用可能かを調べ、
;       16x16 2バイト文字セットが使用可能であれば、
;       変数'read_DBCS_font'に文字パターンを読み取る
;       ための'フォントの読み取り'機能のアドレスが
;       設定されます。
;
mov     ax, 5000h       ;'フォントの読み取り'機能のアドレスを取得
mov     bh, 01h         ;フォントの種類:2バイト文字セット
xor     bl, bl          ;0に設定する
mov     dx, 1010h       ;16x16 フォント
mov     bp, 0           ;コード・ページの省略時値
int     15h
$if     nc, and
 cmp     ah, 0
 $if     e              ;正常に機能は完了
        mov     word ptr read_DBCS_font, bx
        mov     word ptr read_DBCS_font + 2, es

$else
        ; 機能はサポートされていない
        ; エラー・メッセージを表示してプログラムは終了
                          :
                          :

$endif
        :
        :
;
;       目的の2バイト文字セットの文字パターンを読み取ります。
;
mov     cx, ????        ;2バイト文字コード
push    ds
pop     es
lea     si, DBCS_font_buffer
call    read_DBCS_font

        :
        :
```

AH=87H ブロックの移動

この機能は実モードのプログラムやシステムが保護モードに切り替えることによって使用できる保護モードのアドレス範囲である1MB以上の記憶域にデータ・ブロックを転送したり、1MB以上の記憶域からデータ・ブロックをすることができます。

<設定>

(AH) = 87H — ブロックの移動

(CX) 転送するデータ・ブロック（記憶域）のワード・カウント
最大数 = 8000H （32Kワード（64Kバイト））

(ES:SI) この機能を使うルーチンにより構築されるグローバル記述子テーブル(GDT)の位置

(ES:SI)が指し示すグローバル記述子テーブル(GDT)は、この機能を呼び出す前に構築します。記述子テーブルは保護モードでのブロックの移動を実行するために使用されます。転送元と転送先の記述は、ユーザーにより構築され(CX – 1)の2倍以上か、それに等しいセグメント長が必要です。データ・アクセスの右側のバイトは、CPL0–R/W(93H)にセットしなければなりません。24ビット・アドレス（高位にバイト、低位にワード）を転送元と転送先に対してセットしなければなりません。

注: 転送中にどのような割り込みも許可されません。したがって、大容量のブロック転送は割り込みを失う場合があります。

<結果>

(AH) = 00H 命令は正常に完了
= 01H RAMパリティー（パリティー・エラー・レジスター消去）
= 02H 他の例外割り込みエラーが発生
= 03H ゲート・アドレス・ライン20H失敗
(AH)以外のすべてのレジスターは回復されます。

(AH) = 00H の場合
CF = 0
ZF = 1

(AH) = 00H~03H の場合
CF = 1
ZF = 0

ブロック移動のGDTの構成を以下に示します。

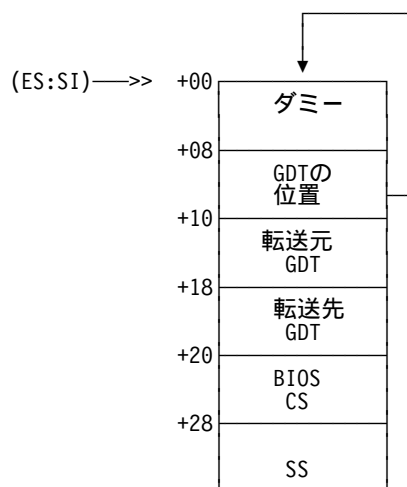



図 6. ブロック移動のグローバル記述子テーブル

定義される記述子テーブルは、以下のとおりです。

- 最初のダミーは必要であり、ユーザーが0に初期設定します。
- ２番目はデータ・セグメントとしてのGDTを指し示します。この領域は、ユーザーが0に初期設定し、BIOSにより修正されます。
- ３番目は転送元を指し示し、ユーザーが初期設定します。
- ４番目は転送先のセグメントを指し示し、ユーザーが初期設定します。
- ５番目は保護モード・コード・セグメントを作成するために、BIOSによって使用されます。
- ６番目は保護モード・スタック・セグメントを作成するために、BIOSによって使用されます。この領域はユーザーが0に初期設定し、BIOSにより修正され、ユーザー・スタックを指し示します。

転送元と転送先の記述子テーブルのサンプルを以下に示します。

```
SOURCE_TARGET_DEF       STRUC

  SEG_LIMIT        DW       ?   ; セグメントの限界値（1~65536バイト）
  LO_WORD          DW       ?   ; 24ビット・セグメント物理
  HI_BYTE          DB       ?   ; アドレス[0~(16MB–1)]
  DATA_ACC_RIGHTS  DB     93H   ; アクセス権バイト(CPL0-R/W)
  予約済み          DW       0   ; 予約済みワード（0に設定）

SOURCE_TARGET_DEF       ENDS
```

グローバル記述子テーブル[(ES:SI)により指し示される実際の位置]

```
BLOCKMOVE_GDT_DEF        STRUC
         DW     0,0,0,0           ; １番目の記述はアクセス不可
CGDT_LOC DW     ?,?,?,0           ; 呼び出しルーチンのGDTの位置
SOURCE   DW     ?,?,?,0           ; 転送元の記述子
TARGET   DW     ?,?,?,0           ; 転送先の記述子
BIOS_CS  DW     ?,?,?,0           ; BIOSコード記述子
TEMP_SS  DW     ?,?,?,0           ; スタック記述子
BLOCKMOVE_GDT_DEF        ENDS
```

AH=88H 拡張メモリー・サイズの判別

このルーチンはアドレス100000H以上のシステム・メモリーの量を返します。

＜結果＞

(AX) アドレス100000H以上の使用可能なメモリーの連続する1KB単位のブロック数

AH=89H ~ FFH 予約済み

INT 16H キーボード入出力

AHの指定に従って以下の機能を実行します。

AH=00H 次文字の読み取り

キーボードでタイプされたデータを読み取ります。AHに走査コード、ALに文字コードを返します。

＜結果＞

(AL) 文字コード
(AH) 走査コード

既に入力されたデータがキーボード・バッファーに残っている場合は、その先頭のデータを返します。キーボード・バッファーが空の場合は、データが入力されるまでこのルーチン内で待ちます。

注:

1. 走査コードと文字コードは、正しい組み合せが見つかるまでキー・バッファーから読み取られます。
2. キーとコードの対応については、「ソフトウェア・プログラミング・ガイド」の“キーボード入力”を参照してください。

このBIOSルーチンにより返される走査コード／文字コードの組み合せの種類を表8に示します。かな漢字変換もこの表に準拠します。

表 8. 走査コード/文字コードの組み合わせの種類

種類	**(AH)**	**(AL)**
1バイト・コード	走査コード	文字コード
1バイト・コード(カタカナ)	x‘00’	文字コード
拡張コード	拡張コード	x‘00’またはx‘E0’
ALT（前面）キー+JIS8ビット・コード入力	x‘00’	文字コード
2バイト・コードの第1バイト	x‘00’	第1バイト
2バイト・コードの第2バイト	x‘00’	第2バイト
2バイト・コード用拡張コード(*1)	2バイト・コード用拡張コード	x‘00’

注:

*1 英数、ひらがな、カタカナ、ローマ字、漢字、漢字番号、半角／全角、変換、無変換の各キーを押したときに生成されますが、入力支援サブシステム ($IAS.SYS) で処理するため、これらのキーは通常読み取ることはきません。

AH=01H 文字の入力状況

キーボードから入力されたデータがキー・バッファー内にあるかどうかを調べます。状況はZF（ゼロ・フラグ）に設定されます。

＜結果＞

ZF　状況

= 1　読み取れるコードがバッファー内にない。

= 0　読み取れるコードがバッファー内にある。

ZF=0の場合、バッファー内にある次に読み取れる走査コードと文字コードがAXに返されます。ただし、このデータはキー・バッファーに残ったままで、次文字の読み取り(AH=00H)によりキー・バッファーから除去されます。

注:

1. 走査コードと文字コードは、正しい組み合せが見つかるまでキー・バッファーから読み取られます。
2. キーとコードの対応については、「プログラミング・リファレンス・ガイド」の“キーボード入力”を参照してください。

AH=02H シフト状況

現在のキーボードのシフト状況を読み取ります。

＜結果＞

(AL)　現在のシフト状況

ビット7 = 1	挿入モード
ビット6 = 1	Caps Lockモード
ビット5 = 1	Num Lockモード
ビット4 = 1	Scroll Lockモード
ビット3 = 1	Altキー押し下げ
ビット2 = 1	Ctrlキー押し下げ
ビット1 = 1	左シフト・キー押し下げ
ビット0 = 1	右シフト・キー押し下げ

(AH)　予約済み

AH=03H キーボード・タイプ速度の設定

＜設定＞

(AL)　= 05H

(BL)　タイプ間隔時間（１秒当たりにタイプできる文字数）

表 9. キーボード・タイプ速度の設定値

設定値	速度	設定値	速度	設定値	速度
00H	30.0	0BH	10.9	16H	4.3
01H	26.7	0CH	10.0	17H	4.0
02H	24.0	0DH	9.2	18H	3.7
03H	21.8	0EH	8.6	19H	3.3
04H	20.0	0FH	8.0	1AH	3.0
05H	18.5	10H	7.5	1BH	2.7
06H	17.1	11H	6.7	1CH	2.5
07H	16.0	12H	6.0	1DH	2.3
08H	15.0	13H	5.5	1EH	2.1
09H	13.3	14H	5.0	1FH	2.0
0AH	12.0	15H	4.6	20H~FFH	予約済み

(BH) 遅延時間（ミリ秒）

= 00H 250
= 01H 500
= 02H 750
= 03H 1000
04H~FFH 予約済み

AH=04H 予約済み

AH=05H キー・バッファーへの書き込み

キーボードから入力されたものと同様に、キー・バッファーに走査コードと文字コードを書き込みます。

＜設定＞
(CL) 文字コード
(CH) 走査コード

＜結果＞
(AL) 完了コード
= 00H 機能は正常に終了
= 01H バッファーが一杯

AH=06H~0FH 予約済み

AH=10H 次文字の読み取り（拡張機能）

キーボードでタイプされたデータを読み取ります。機能は、‘AH=00H 次文字の読み取り’と同じです。ただし、読み取られる走査コード／文字コードの一部が、異なります。走査コードと文字コードについては、「プログラミング・リファレンス」の“キーボード入力”を参照してください。

AH=11H 文字の入力状況（拡張機能）

この機能は、‘AH=01H 文字の入力状況’と同じです。ただし、返される走査コード／文字コードが、一部異なります。走査コードと文字コードについては、「プログラミング・リファレンス」の“キーボード入力”を参照してください。

AH=12H シフト状況（拡張機能）

機能は、‘AH=02H シフト状況’と同じです。

＜結果＞

(AL) 現在のシフト状況

ビット7 = 1	挿入モード
ビット6 = 1	Caps Lockモード
ビット5 = 1	Num Lockモード
ビット4 = 1	Scroll Lockモード
ビット3 = 1	Altキー押し下げ
ビット2 = 1	Ctrlキー押し下げ
ビット1 = 1	左シフト・キー押し下げ
ビット0 = 1	右シフト・キー押し下げ

(AH) 拡張シフト状況

ビット7 = 1	Sys Rqキー押し下げ
ビット6 = 1	Caps Lockキー押し下げ
ビット5 = 1	Num Lockキー押し下げ
ビット4 = 1	Scroll Lockキー押し下げ
ビット3 = 1	右側のAltキー押し下げ
ビット2 = 1	右側のCtrlキー押し下げ
ビット1 = 1	左側のAltキー押し下げ
ビット0 = 1	左側のCtrlキー押し下げ

AH=13H ２バイト文字セットの状況モードの設定と読み取り

2バイト文字セットの状況モードを設定（変更）したり、読み取ることができます。

＜設定＞

(AL) 設定または読み取り
= 00H 2バイト文字セットのキーボード状況モードを設定します。
= 01H 2バイト文字セットのキーボード状況モードを読み取ります。

(DH) 予約済み（変更しないでください）

(DL) 状況モード

ビット7 = 0	非漢字モード
= 1	漢字モード

ビット6 = 0	ローマ字オフ
= 1	ローマ字オン
ビット5~3	予約済み
ビット2, 1 = 00	英数シフト
= 01	カタカナ・シフト
= 10	ひらがなシフト
= 11	予約済み
ビット0 = 0	半角
= 1	全角

注: この機能は、入力支援サブシステム($IAS.SYS)により提供されます。入力支援サブシステムが導入されていない場合は、BIOSは省略時のシフト状況(DX=0)を返します。ただし、かな漢字変換プログラム（たとえば、$IAESKK.SYS）が導入されていないと、ビット7は無効（常に0）です。

AH=14H シフト状況の表示と消去

画面最下行のキーボード・シフト状況の表示および消去を実行します。

＜設定＞

(AL)

= 00H シフト状況の表示
= 01H シフト状況の消去
= 02H シフト状況の表示の有無を読み取る

＜結果＞ AL = 02Hの場合に返されます。

(AL) シフト状況の表示状態
= 00H 表示
= 01H 消去

注: この機能は、入力支援サブシステムの機能の一部です。入力支援サブシステムが導入されていなければ、BIOSは、常に省略時のシフト状況の表示状態（消去）を返します。

INT 17H プリンター入出力

プリンターBIOSでは、論理プリンター・ポート(LPTx)をレジスターDXで指定します。最大3個までの論理プリンター・ポートを使用できます。

LPT1 －最初に見つかった物理プリンター・ポート(パラレル1～3）
LPT2 －２番目に 〃
LPT3 －３番目に 〃

たとえばシステム・ボード上のプリンター・ポートをパラレル2と指定しても、DOS, BIOSなどではLPT1として扱われます。

(DX)=0, 1, 2で論理プリンター・ポートLPT1, LPT2, LPT3を指定します。

論理プリンター・ポートLPT1, LPT2, LPT3に対応する物理プリンター・ポートのアドレスは、それぞれBIOSデータ域40:08, 40:0A, 40:0Cに格納されています。

プリンターBIOSでは、プリンター・ポートのバイト・ストリーム入出力機能を以下のパラメーターに従って提供します。

AH=00H 文字の印刷

プリンターに印刷する文字または制御コードをALに指定します。 16進の00~1Fおよび7Fを指定すると、特殊な文字が印刷されます。制御コードについては、プリンターにより処理が異なるため、プリンターの説明書を参照してください。

＜設定＞

(AL) 印刷する文字
(DX) 論理プリンター・ポート(0, 1, 2)

＜結果＞

(AH) 状況

印刷終了時、AHに状況が返されます。状況については、『AH=02H 状況の読み取り』を参照してください。

注:

1. 印刷する文字が全角の場合、2バイト・コードの第1バイトに続けて、第2バイトを指定します。第2バイト目を受け取った時点で印字されます。第1バイト目と第2バイト目の文字印刷の間に他のデータ（文字やイメージ）の印刷を実行しないでください。その場合、結果は保証されません。

2. ユーザー定義文字が指定された場合、対応する文字イメージを取得し、そのイメージをプリンターに送信します（この機能に対応するプリンター・ドライバーが導入されている場合）。

3. ESC制御コード列内であれば、指定されたコードがそのままプリンターに送信されます。従って、ESC制御コード列内にユーザー定義文字に相当するデータが含まれていても変換されることなくそのままプリンターに送信されます。

AH=01H プリンター・ポートの初期設定

ハードウェアの初期設定、ソフトウェア状況のリセット、初期制御値の設定などを行います。初期制御値については、各プリンターの説明書、技術解説書などを参照してください。

＜設定＞

(DX) 論理プリンター・ポート(0, 1, 2)

＜結果＞

(AH) 状況

印刷終了時、AHに状況が返されます。状況については、『AH=02H　状況の読み取り』を参照してください。

AH=02H 状況の読み取り

＜設定＞

(DX) 論理プリンター・ポート(0, 1, 2)

＜結果＞

(AH) 次の状況が返されます。

ビット7 = 1	作動可能
ビット6 = 1	要求した命令に対する応答(返事)
ビット5 = 1	用紙切れ、または、自動給紙機構中の用紙ジャム
ビット4 = 1	選択（オンライン）
ビット3 = 1	I/Oエラー。
ビット2~1	予約済み
ビット0 = 1	タイム・アウト。印刷したい文字や送りたい制御コードがあるのに、使用中の状態が一定時間を越えて続いた場合に返されます。

INT 18H～19H BIOS用に予約済み

これらの割り込みは、BIOS用に予約済みです。

INT 1AH 時刻

このルーチンは、AHの指定に従って、時刻を設定したり、読み取ったりします。

注: システム・タイマーとリアル・タイム・クロックは、両方を同時に設定する必要があるため、以下の時刻や日付の設定(AH=01H, AH=03H, AH=05H)を実行する代わりに、DOS機能呼び出しINT21HのAH=2A, 2B, 2C, 2Dを使って時刻・日付を設定してください。

AH=00H システム・タイマーの時刻カウントの読み取り

以下の情報を返します。

(CX) カウントの高位部分
(DX) カウントの低位部分
(AL) = 0 タイマーが最後に読まれてから24時（午前０時）を経過していない。
<>0 24時を経過している。

AH=01H システム・タイマーの時刻カウントの設定

以下の情報を設定します。

CX カウントの高位部分
DX カウントの低位部分

注: カウントは、1193180/65536回/秒、すなわち、毎秒約18.2回の割合で増えます。

AH=02H リアル・タイム・クロックの時刻の読み取り

以下の情報を返します。

(CH) 時（２進化10進数(BCD)）
(CL) 分（２進化10進数(BCD)）
(DH) 秒（２進化10進数(BCD)）
(DL) = 01H 夏時間制選択
= 00H 夏時間制なし

CF = 0 クロックは作動している
CF = 1 クロックが作動していない

AH=03H リアル・タイム・クロックの時刻の設定

以下の情報を設定します。

(CH) 時（２進化10進数(BCD)）
(CL) 分（２進化10進数(BCD)）
(DH) 秒（２進化10進数(BCD)）
(DL) = 01H 夏時間制選択
= 00H 夏時間制なし

AH=04H　リアル・タイム・クロックの日付の読み取り

以下の情報を返します。

(CH)　世紀（２進化10進数(BCD)で19または20）
(CL)　年（２進化10進数(BCD)）
(DH)　月（２進化10進数(BCD)）
(DL)　日（２進化10進数(BCD)）

CF = 0　クロックは作動している
CF = 1　クロックが作動していない

AH=05H　リアル・タイム・クロックの日付の設定

以下の情報を設定します。

(CH)　世紀（２進化10進数(BCD)で19または20を指定）
(CL)　年（２進化10進数(BCD)）
(DH)　月（２進化10進数(BCD)）
(DL)　日（２進化10進数(BCD)）

AH=06H~FFH　予約済み

INT 1BH キーボードBREAKアドレス

キーボードBREAKアドレスを含みます。

INT 1CH タイマー刻みアドレス

タイマー刻みタイマーを含みます。

INT 1DH～1FH BIOS用に予約済み

これらの割り込みは、BIOS用に予約済みです。

第3章　プログラム開発上の考慮点

アプリケーション・プログラムの互換性を保ち、機器の違いに影響されないように、ハードウェアとのインターフェースはできる限りBIOS経由で行うようにしてください。

ハードウェア割り込み

レベル・センス割り込みのシステムでは、ISRの状態をI/Oアドレスのビット位置で読み取り可能です。読み取りは割り込みサービス・ルーチン期間中に行われ、ラッチは読み取り操作によりリセットされるか、または強制的なリセットが必要です。

注:　効率や将来の拡張性を考慮して割り込みレベルを共用する装置の数を抑えることができます。

レベル・センス割り込みのシステムの割り込みコントローラーでは、EOIが送られたときの割り込み要求はインアクティブでなければなりません。そうでないと、"新規の"割り込み要求が検知され、別の割り込みが引き起こされてしまいます。

この問題を避けるためには、レベル・センス割り込みハンドラーが割り込み状態を(通常、割り込みを引き起こした装置のI/Oポートのリード/ライトにより)クリアーしなければなりません。割り込み状態をクリアーした後、割り込みコントローラーへEOIを送る前に必ず**JMP　$+2**命令を実行する必要があります。これは割り込みコントローラーを再度イネーブルする前に割り込み要求がリセットされた状態にしておくためです。EOIを送った後割り込みフラグ・セット(STI)命令で割り込みをイネーブルする前にも別の**JMP　$+2**命令が必要です。

STI命令の直前にI/Oコマンドがある場合、システム・ボードやチャネルの操作によっては割り込みコントローラーの回復時間を十分に確保できないことがあるので、I/OコマンドとSTI命令の間に必ず**JMP　$+2**命令を挿入する必要があります。

注:

1. **MOV AL,AH** タイプの命令では割り込みコントローラーの回復のために十分な時間がとれません。正しい手順の一例を示しておきます。

```
OUT   IO_ADD,AL
JMP   SHORT $+2
MOV   AL,AH
STI
```

2. 割り込みコントローラーをプログラムする場合は、割り込みフラグ・クリアー(CLI)コマンドを発行して割り込みをディセーブルしておく必要があります。これには、マスク・レジスター、EOI, ICW, OCWが含まれています。

レベル・センス割り込みのシステムでは、割り込みコントローラーがエッジ・トリガー・モードに設定されないよう、ハードウェア的に設定されます。

ハードウェア割り込みIRQ9は、カスケード接続でのレベルIRQ2の代替レベルと定義されています。割り込み共用はIRQ2, INT0AHで行うようにプログラムしてください。マイクロチャネル・モデル以外のモデルで使われているIRQ2との互換性を保つために、以下の処理が行われます。

1. ある装置がチャネルのIRQ2上で割り込み要求をアクティブにする。

2. その割り込み要求は、ハードウェア内で2番目の割り込みコントローラー(スレーブ)に対するIRQ9入力としてマッピングされる。

3. 割り込みが発生すると、システムMPUがIRQ9(INT71H)の割り込み処理ルーチンに制御を渡す。

4. この割り込み処理ルーチンは、2番目の割り込みコントローラー(スレーブ)にEOIを発行し、制御をIRQ2(INT0AH)割り込み処理ルーチンに渡す。

5. IRQ2割り込み処理ルーチンは、IRQ2要求の処理を完了したマスター割り込みコントローラーに対してEOIを送る前に、割り込み装置により割り込み要求をクリアーさせる。

BIOSおよびDOSファンクション・コール

ハードウェアになるべく依存しないようにするためには、I/O操作はすべてDOSファンクション・コールを介して実行する必要があります。必要な機能が用意されていない場合は、BIOS割り込みを使ってハードウェアをアクセスしてください。

プログラム作成時は、以下の点を考慮してください。

- 環境によっては、DOS機能をアクセスするためにプログラムによる割り込みが使われます。この方法を使えば、プログラムから絶対アドレッシングをなくすることができます。必要なものは割り込み番号だけです。
- DOSはハードウェアに対する依存性をマスクすることができます。たとえば新しい装置が導入された場合、BIOSを変更し、新しい装置でも従来と同じプログラミング・インターフェースを受け入れられるようにすることができます。
- BIOSにパラメーター・テーブル、たとえばディスプレイやフロッピーディスク用のBPBなど、が用意されている場合、プログラムでテーブルのコピーを新たに作成し、そのコピーを指すようにベクトルを書き替えて新しいパラメーター値と差し替えることができます。プログラムではまず現行のベクトルを使って現行テーブルをコピーし、次にテーブル中の必要箇所を変更する必要があります。このようにすれば、そのまま残しておかなければならない値を間違って変えるということはありません。

システム環境の識別に関するプログラミング上の考慮事項

１バイト文字モードと２バイト文字モード

DOS/Vは英語モードと日本語モードにシステムを切り替えることができます。アプリケーション・プログラム（デバイス・ドライバーとTSR（終了後常駐プログラム）を含む）が、現在作動しているモードを識別する必要がある場合、DOSの２バイト文字セットのベクターを調べる必要があります。次の機能呼び出しにより返されるfarポインターでベクターを参照できます。

INT21H, (AX)=6300H　DBCSベクターの取得

Entry:　なし
Exit:　DS:SI - DBCSベクターのアドレス
DBCSベクター:

```
db      開始１，　終了１
db      開始２，　終了２
 :          :         :
db          0,        0,       ; 終了マーカー
```

どちらのモードか簡単に判別できます。開始１と終了１が０の場合、英語モードです。そうでなければ、日本語モードです。

通常、デバイス・ドライバーおよびTSRは、それらがDOS内部データ域を入れ替えない限り、デバイス・ドライバーおよびTSRの実行中にDOS機能呼び出しを発行できません。このようなプログラムは、導入時にDBCSベクターのポインターを獲得し、必要なときにそのポインターを使用してDBCSベクターを参照します。

コードページを取得することによって日本語 (環境）か、その他の環境かを判定する方法があります。

```
INT21H,(AX)=6601H    コードページの取得
MOV   AX,6601H    ;コードページの取得
INT   21H
JC    ERROR       ;エラーの判定
CMP   BX,932      ;日本語環境？
JNZ   US_ENV
```

JP_ENV:

日本語環境の処置

US_ENV:

その他の環境の処理

また、バイリンガル・サポート・ファンクションを使用した方法もあります。

```
INT2FH,(AX)=4F01H    動作モードの取得
MOV   AX,4F01H
INT   2FH
CMP   BX,932
JNE   US_MODE
```

JP_MODE:

日本語モードの処理

US_MODE:

US モードの処理

注: この機能はBILING.SYSの機能の一部です。BILING.SYSが導入されていなければ使用できません。

第4章 ハードディスク情報

構造

- 複数のオペレーティング・システムで共有できるように、ハード・ディスクは論理的に4つまでの区画に分割できます。区画は連続した空間であり、特定のオペレーティング・システムが1つまたはそれ以上の区画を専有することができます。

  区画の数とその容量は、FDISKコマンドを使ってユーザーが選択できます。FDISKコマンドにより作成された区画の情報は、ハード・ディスクの先頭セクターにある「**マスター・ハード・ディスク・ブート・レコード**」内の「**区画テーブル**」に保持されます。

- どのオペレーティング・システムも、それが収められた区画を1つのディスクとして考え、その機能やユーティリティーが他の区画をアクセスしないようにしなければなりません。

- 各区画内には、その先頭セクターにブート・レコードを含むことができる他、オペレーティング・システムのコピーも含めてプログラムやデータを収めることができます。たとえば、DOSのFORMATコマンドで、フロッピーディスクの場合と同様にDOS用区画をフォーマットする(かつDOSシステムのコピーを書き込む)ことができます。また、FDISKコマンドを使って、ある区画を**ブート可能**(活動状態)に指定することができます。活動状態に指定された区画は、「マスター・ハードディスク・ブート・レコード」により、システムの始動時、または再始動時に制御を受け取ることができます。

システム始動のメカニズム

システム始動のメカニズム(ブート手順)は、以下のとおりです。

1. まず最初にドライブAからシステムをロードします。ドライブの準備ができていない、または読み取りエラーが起きた場合、ハード・ディスク(複数台を装備している場合はその1台目)の先頭セクターから「マスター・ハードディスク・ブート・レコード」を読み取ります。
2. ハードディスクからの読み取りが成功した場合、「マスター・ハードディスク・ブート・レコード」に制御が渡され、その中の「区画テーブル」が調べられます。**ブート可能な** (活動状態の)区画を示す項目があれば、その区画のブート・レコードが(その区画の先頭セクターから)読み取られ、制御が渡されます。
3. 区画テーブル中に無効な「ブート標識」(ブート・インディケーター)があったり、**ブート可能**に設定されている「ブート標識」が2つ以上あると、**区画テーブルが無効**であることを知らせるメッセージが表示され、システムはイネーブル・ループに入ります。この場合、システム・フロッピーディスクをドライブAに入れ、システム・リセットによりフロッピーディスクからDOSを始動することができます。
4. 区画のブート・レコードが5回の再試行後も読み取りエラーによりうまく読み取れない場合、**オペレーティング・システムのロード・エラー**を知らせるメッセージが表示され、システムはイネーブル・ループに入ります。
5. 区画のブート・レコードに有効な「署名」がなかった場合、**オペレーティング・システムがない**ことを知らせるメッセージが表示され、システムはイネーブル・ループに入ります。

ブート・レコードについては次ページ以降に説明があります。

注: 区画のサイズや場所を変更するときは、ハード・ディスク中の全ファイルをバックアップしておく必要があります(区画設定作業により、それまでの区画境界のトラックが失われます)。

ブート・レコードの区画テーブル

ハードディスク・ブート・レコードは、ハードディスク(複数台ある場合はそのすべて)の先頭セクターまたは拡張区画内の各論理ディスクの先頭セクターに記入されている必要があります。この中には以下の内容を含んでいます。

- **コード**:　最大4つまであるオペレーティング・システムの中からいずれか1つのブート・レコードをロードし、制御を渡します。
- **区画テーブル**:　ブート・レコードの終りに位置します。各区画テーブルは16バイト長で、以下のものを含んでいます(次ページの図を参照)。
  - 区画の開始/終了シリンダー
  - 区画の開始/終了セクター
  - 区画の開始/終了ヘッド
  - 区画に先行するセクター数
  - 区画の占めるセクター数

  ブート標識バイトは、その区画にロード可能なオペレーティング・システムが含まれているかどうかを判断するために使われます。FDISKコマンドは、この標識に80Hを(かつ他の区画のブート標識にはすべて00Hを)セットして、その区画が**ブート可能**である旨の印を付けます。

  いずれかのブート標識に80Hがセットされていると、標準のブート・ルーチンがブート標識に続く3バイトで指示されるセクターをロードします。このセクターが、選択されたオペレーティング・システムの実際の(本当の)ブート・レコードで、これ以降のシステム・ロードはすべてこのブート・レコードが責任を持つことになります。ブート・レコードはすべて絶対番地 0:7C00 にロードされます。

区画テーブル

↑
ブート・レコード
(先頭セクター内)

ハードディスク
または
論理ディスク

ブート・レコード内のオフセット					
0					
1BE	区画#1 開始	ブート標識	H	S	CYL
1C2	〃 終了	システム標識	H	S	CYL
1C6	〃 相対セクター	低位ワード		高位ワード	
1CA	〃 セクター数	低位ワード		高位ワード	
1CE	区画#2 開始	ブート標識	H	S	CYL
1D2	〃 終了	システム標識	H	S	CYL
1D6	〃 相対セクター	低位ワード		高位ワード	
1DA	〃 セクター数	低位ワード		高位ワード	
1DE	区画#3 開始	ブート標識	H	S	CYL
1E2	〃 終了	システム標識	H	S	CYL
1E6	〃 相対セクター	低位ワード		高位ワード	
1EA	〃 セクター数	低位ワード		高位ワード	
1EE	区画#4 開始	ブート標識	H	S	CYL
1F2	〃 終了	システム標識	H	S	CYL
1F6	〃 相対セクター	低位ワード		高位ワード	
1FA	〃 セクター数	低位ワード		高位ワード	
1FE	署名				

図 7. 区画テーブル

ブート標識:

00H: ブートできない区画
80H: ブートできる区画

複数の区画に80Hをセットすることはできません。

システム標識:　区画を専有するオペレーティング・システムの標識、または拡張区画の標識を含む。DOS/Vの場合は次のとおり。

00H: 未定義
01H: 基本DOS(12ビットFAT)
04H: 基本DOS(16ビットFAT:32MB以下)
05H: 拡張DOS
06H: 基本DOS(16ビットFAT:32MB超)

シリンダー(CYL)およびセクター(S)：

(CYL)：　シリンダー番号の低位8ビット
(S)の
高位２ビット：　シリンダー番号の高位2ビット

この10ビットで示されるシリンダー番号は、INT13H（ディスク入出力)でCH, CLにセットするシリンダー番号に対応しています。

上記各フィールドは、MOV命令を2回発行するだけでDXおよびCXレジスターを設定し、BIOS割り込みを発行して適切なブート・レコードをロードできるように配置されています（ハード・ディスクからのブートは、1台目のハードディスクからのみ可能です。BIOS割り込み時DLに設定するドライブ番号(80H)はブート標識に対応しています）。

区画はすべてセクター1, ヘッド0から始まり、シリンダー境界に整列するように配置されます。

例外）　ハードディスクの先頭に配置される区画は、シリンダー0, ヘッド1, セクター1から始まる必要があります。これは、そのハードディスクの「マスター・ハードディスク・ブート・レコード」およびハードディスク・タイプ定義用の情報を収めるスペースを残しておくためです。オペレーティング・システムがシリンダー0, ヘッド0を使うことはできません。

相対セクター:　4バイトのフィールドで、各区画の先行セクター数が保持されています。この値は、シリンダー0, セクター1, ヘッド0から始めて、セクター、ヘッド、トラックの順に計算して求めます。したがって、ディスクが1トラック当たり17セクターで4ヘッドの場合で、2番目の区画がシリンダー1, セクター1, ヘッド0から始まっていると、2番目の区画の**相対セクター**は68(10進数)となります。つまり、その区画よりも前に4ヘッド17セクターずつのトラックが1個存在するということです。このフィールドは、低位ワードから先に保管されます。

セクター数:　区画に割り振られたセクター数が保持されています。低位ワードから先に保管された4バイトのフィールドです。

署名:　ブート・レコードの末尾の2バイト(55AAH)は、有効なブート・レコードを識別するための**署名**として使われます。区画テーブルの他、各区画のブート・レコードにも、オフセット1FEHに署名が必要です。

ある区画のブート・レコードに制御が渡されると、DS:SIに区画テーブルの入口アドレスが渡されます。

拡張DOS区画

「拡張DOS区画」は、大容量のディスクを有効に利用するために用意された区画タイプです。マスター・ハードディスク・ブート・レコードの区画テーブルにシステム標識05Hがあると、拡張DOS区画であることを意味しています。この区画はブート可能ではないので、ブート区画を設定できるプログラム(たとえばDOSのFDISK)ではこの区画をブート可能にできないようにしておく必要があります。

拡張DOS区画は、ブート可能なドライブにすでに基本DOS区画がある場合にのみ作成することができます。基本DOS区画のシステム標識は、01H, 04H, または06Hです。ドライブがブート可能でない場合、基本DOS区画なしでも拡張DOS区画を作成することができます。

拡張DOS区画も、他の区画と同様にシリンダーの境界で開始/終了します。

拡張DOS区画の構造

拡張DOS区画は、ポインターによりリンクされた拡張ボリュームの集まりです。拡張ボリュームは、拡張ブート・レコードと論理ブロック装置1個から構成され、この拡張ブート・レコードにポインターが記述されています。

拡張DOS区画内に作成された拡張ボリュームのサイズは、1シリンダーから拡張DOS区画内の連続した最大使用可能空間までの任意の値をとることができます。

拡張ボリュームもシリンダーの境界で開始/終了しなければなりません。拡張ボリュームは、物理的なディスク・イメージと対応し、拡張ブート・レコードは実際の物理ディスクの先頭にあるマスター・ブート・レコードに対応しています。

また、論理ブロック装置はマスター・ブート・レコードにより指示されるDOS区画に対応しています。

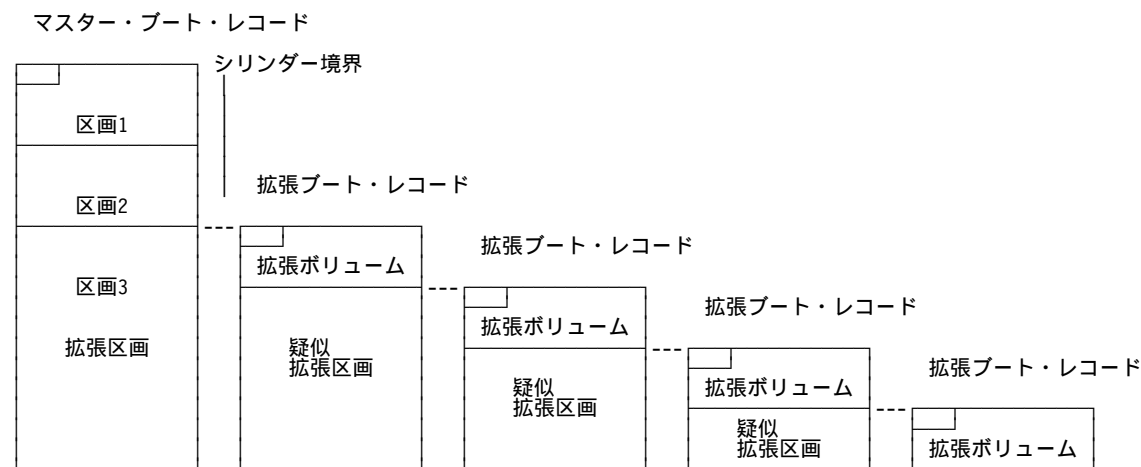



図 8. 拡張DOS区画の構造

したがって、その論理ブロック装置がDOS論理ブロック装置の場合（システム標識が01H, 04Hまたは06H)ふつうのDOSブート・セクターから始まることになります。

この論理ブロック装置は、物理ディスク上の拡張ブート・レコードに続けて、トラック境界から始まる必要があります。論理ブロック装置および拡張ボリュームは共に同じシリンダー境界で終わります。

拡張区画ブート・レコード

各拡張ボリュームは、そのボリュームに割り当てられたディスク位置の先頭セクターに拡張ブート・レコードを持っています(79ページの図8)。この拡張ブート・レコードには55AAHの**署名**バイトを含んでいます。これにより、拡張(またはマスター)ブート・レコードを見るプログラムの互換性が保たれることになります。この拡張ブート・レコードには、さらに「論理ドライブ・テーブル」が含まれており、このテーブルには2種類の項目を含むことができます。

拡張DOS区画のブート・レコードは、マスター・ブート・レコードと似通っています。主な違いは、拡張DOS区画用のブート・レコードには区画テーブルの代わりに論理ドライブ・テーブルが含まれていること、そしてシステム標識05Hが拡張区画ではなく次の論理ドライブを指していることです。

拡張区画ブート・レコードの論理ドライブ・テーブル

拡張区画ブート・レコード中の論理ドライブ・テーブルは、構造上はマスタ・ブート・レコード中の区画テーブルと同じです。テーブルには16バイト長の項目が4つ含まれています。

システム標識バイトには、4つの項目すべてについて次のいずれかの値が記入されていなければなりません。

- 00H: 割り振り空間なし
- 01H: 基本DOS(12ビットFAT)
- 04H: 基本DOS(16ビットFAT:32MB以下)
- 05H: 次の拡張ボリュームに割り当てられた領域に対応し、次の拡張ブート・レコードへのポインターの役割を果たす。
- 06H: 基本DOS(16ビットFAT:32MB超)

システム標識が00Hの場合、その項目の値はすべて0でなければなりません。

ドライブ開始/終了フィールド(CYL, H, S)は、システム標識が記入されている項目すべてに対して指定が必要です。これは、FDISKなどのプログラムが拡張DOS区画に割り振られた空間を判定し、デバイス・ドライバーが自分用のディスク領域を知ることができるようにするためです。

論理ブロック装置を指す項目(システム標識01H, 04H, 06H)の場合、ドライブ開始/終了フィールド(CYL, H, S)は論理ブロック装置の物理境界に対応し、拡張ブート・レコードの先頭からのオフセットを示します。また、次の論理ブロック装置を指す項目(システム標識05H)の場合、ドライブ開始/終了フィールド(CYL, H, S)は次の論理ブロック装置の物理境界に対応し、ハード・ディスクそのものの先頭からのオフセットを示します。

相対セクターフィールドと**セクター数**フィールドは、その項目のシステム標識により設定は異なります。システム標識に01H, 04H, 06Hが設定されている場合、相対セクターには該当する拡張ボリューム用の拡張ブート・レコードの先頭から(先頭を含む)論理ブロック装置の先頭までのオフセットを指定しなければなりません。セクター数フィールドには、作成された論理ブロック装置領域(言い換えれば、開始/終了CYL/H/Sに対応する領域の占めるセクター数)を指定します。拡張ボリューム全体のサイズは、関連する拡張ブート・レコードの相対セクター・フィールドと、セクター数フィールドを加えて求めることができます。

システム標識が05Hの場合、相対セクター・フィールドは拡張DOS区画の先頭から(次の拡張ボリュームまで)のオフセットです。セクター数フィールドは使用されず、00Hを記入しておく必要があります。

構造的に拡張ブート・レコード1つに対して1個の論理ブロック装置だけしか定義できないようになっています。したがって、拡張ブート・レコード内には最大2個の区画項目、つまりシステム標識が01H, 04H, 06Hのいずれかである項目と、次の拡張ボリュームへのポインターであるシステム標識05Hの項目だけが使われます。使用できる項目

の数は2つだけですが、この装置を導入するプログラムで最初の2項目が必ずゼロ以外であると仮定することはできません。

拡張ブート・レコードの最後の2バイト(55AAH)は、そのブート・レコードが有効であることを識別するために使われます。このレコードともに論理ドライブのブート・レコードにもオフセット1FEHにこの署名が必要です。

ハードディスクの割り振り算出方法

DOSは以下の算出式によりハードディスク空間の割り振りを決定します。

$$SPF = \frac{TS - RS - \frac{D*BPD}{BPS}}{CF + \frac{BPS*SPC}{BPC}}$$

TS: ディスクの総セクター数

RS: ブート・レコード用予約セクター数。
日本語DOSの場合は1セクター。

D: ルート・ディレクトリーのディレクトリー項目数。

BPD: レクトリー項目1個当たりのバイト数。
BPBは常に32。

BPS: 論理セクター当たりのバイト数。
一般的にBPSは512バイトだが、RAMDRIVEで他のサイズを指定することができる。

CF: 1ディスク当たりのFAT数。
ふつうは2だが、RAMDRIVEでは1。

SPF: FAT当たりのセクター数。最大値は64。

SPC: 1割り振り単位のセクター数。

BPC: 1FAT項目当たりのバイト数。
12ビットFATシステムでは1.5,
16ビットFATシステムでは2。

付録A. 表示画面の定義とAPAバッファー

ディスプレイの種類とモード

ディスプレイ・モードの設定方法と各モードに対するパラメーターは、下表のとおりです。

ディスプレイ・モード	設定方法	
	BIOS割り込み INT 10H AH=0を送る*1	**ANSI.SYSに ESC［=nh列を送る*2**
80×25 16色カラー文字	(AL)=03H	n=3
640×480(80×30) 2色カラー・グラフィック	(AL)=11H	n=17
640×480(80×30) 16色カラー・グラフィック	(AL)=12H	n=18
80×25 拡張16色カラー文字	(AL)=73H	n=115
640×480(80×25) 16色カラー・グラフィック	(AL)=72H	n=114

***1** マクロ・アセンブラー言語で書かれたプログラムからモード設定するときに便利です。

***2** 各種コンパイラー言語で書かれたプログラムからモード設定するときに便利です。CONFIG.SYSファイルに DEVICE=ANSI.SYSコマンドを含める必要があります。

画面の定義

各モードで、画面に表示される文字またはドットの数は以下のとおりです。

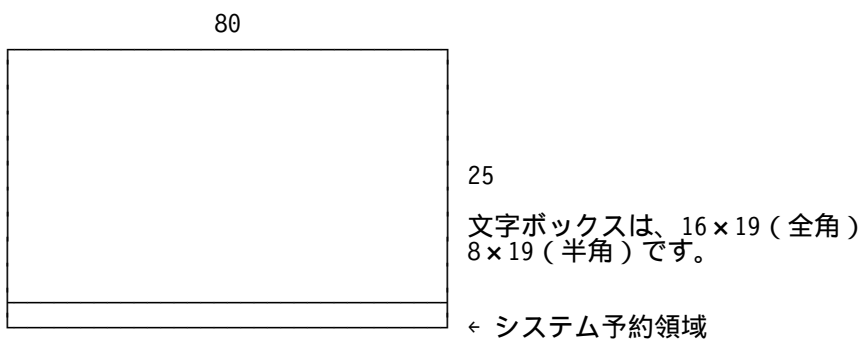
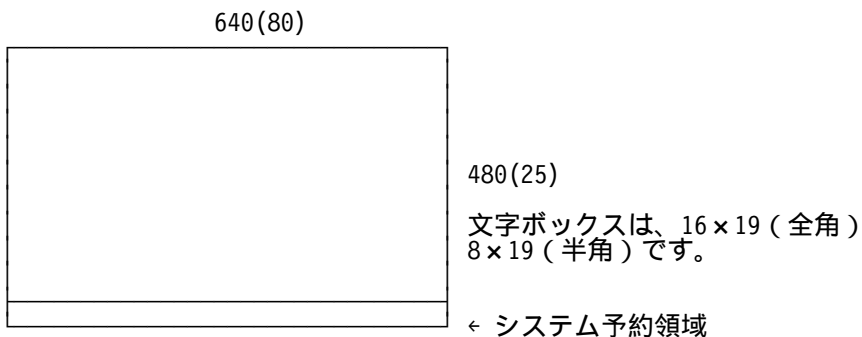
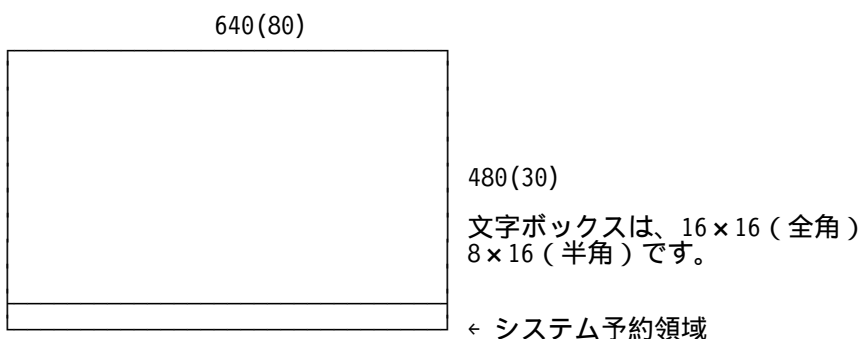
注: どのモードでも、最下行（25行目または30行目）はシステム用に予約されているため、ユーザーは使用できません。使用された場合、結果は予測できません。

文字モード（ビデオモード03H, 73H）

80

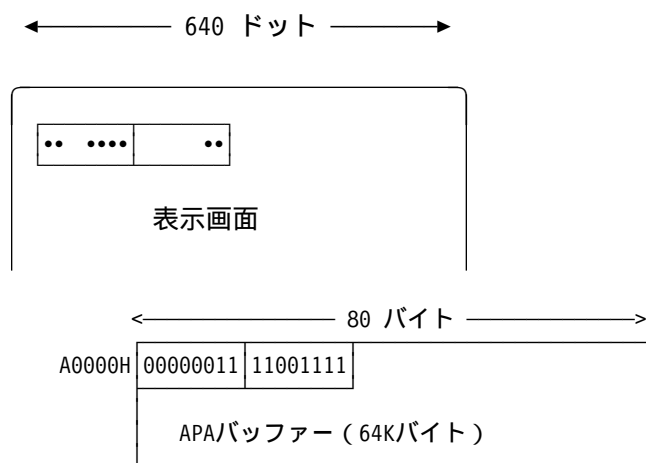
25

文字ボックスは、16×19（全角）
8×19（半角）です。

← システム予約領域

640×480ドット・グラフィック・モード（ビデオモード11H, 12H）

カッコ内は、グラフィック・モードにおける文字表示数（行および桁）を示します。文字は、80字×30行で表示されます。
文字ボックスは、16×16（全角）、8×16（半角）です。

APAバッファーの構造

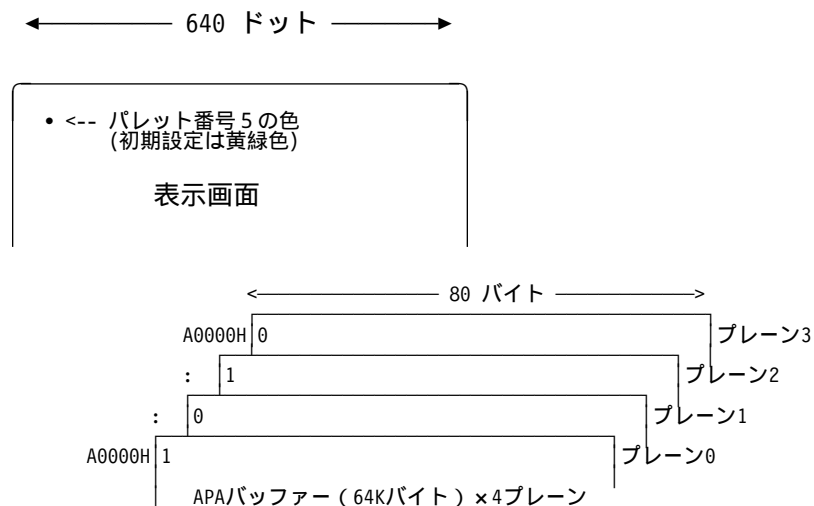
640×480 ２色グラフィック・モード

表示画面の各ドットは、システム記憶域の絶対アドレスA0000Hから始まる全点アドレス可能(APA)バッファー(64Kバイト)の各ビットと対応しています。

APAバッファーの80バイトが表示画面の１つのドット行に対応し、80バイトの各ビットが画面の各ドットに対応しています。（80×8=640ドット）ただし、ビットとドットはワード単位で対応しており、低アドレスのバイトが画面上で左の８ドットに対応し、高アドレスが画面上の右の８ドットに対応しています。１バイトの中では、高位ビットが画面上の左のドットに対応しています。各ビットを１に設定すると対応するビットが光ります。APAバッファーのサイズは64Kバイトですが、そのうち38,400(=80x480)バイトが実際に使用されます。

ユーザー・プログラムは、APAバッファーを直接アクセスすることにより、BIOSを経由しないで画面ドットを読み書きすることができます。

注: APAバッファー域(64Kバイト)を超える領域に書き込みを行わないでください。書き込みが行われた場合、その結果は予測できません。

640×480 16色カラー・グラフィック・モード

表示画面の各ドットの表示色は、システム記憶域の絶対アドレスA0000Hから始まる4枚の全点アドレス可能(APA)バッファー(64Kバイト)の各ビットにより選択されるパレット番号に設定された色となります。

パレットは0から15までの16個の中から選択できます。各パレットに設定できる色の値は6ビット値ですから64色の中のいずれか1色をパレットに設定することができます（INT10HのAH=10H）。

注: グラフィック・カーソルはアプリケーション・プログラムにおいて独自にAPAバッファー上に書き込む必要があります。

テクニカル・リファレンス

DOS/V マウス・ドライバー

技術解説編

特記事項

「DOS/V　マウス・ドライバー　技術解説編」は、インター・ナショナル・ビジネス・マシーンズ・コーポレーション（以下IBMと略します）の著作物です。これはIBM社の出版物「マウス・ドライバー　ユーザーズ・ガイド(DOS/V用)」の内容を記載しており、この著作権はIBMが所有しています。この技術解説編にあるアセンブラーで組まれたサンプル・プログラムは、すべてOADGの著作物です。

これらの内容の無断転載、複製などの著作権に抵触する使用はできません。

目次

第1章　マウス・ドライバーを使用できるようにするには

この章では、マウス・ドライバーを使えるようにするために必要な手順を説明します。

マウス・ドライバーを読み込む前に、マウスが正しく接続されていることを確認してください。

マウス・ドライバーを使用できるようにするには,記憶域に読み込まなければなりません。DOSディスケット、またはハード・ディスクに入っているMOUSE.COMプログラムを実行するとマウス・ドライバーをシステム・ユニットの記憶域に読み込むことができます。実際の手順は次のとおりです。

MOUSE.COMがディスケットに入っている場合

1. MOUSE.COMファイルが入ったディスケットをドライブAに差し込んでください。
2. MOUSE.COMが入っているディレクトリーを現行ディレクトリーにしてください。
3. MOUSEとタイプしてEnter(改行)キーを押してください。

MOUSE.COMがハード・ディスクに入っている場合

1. MOUSE.COMが入っているディレクトリーを現行ディレクトリーにしてください。
2. MOUSEとタイプしてEnter(改行）キーを押してください。

これでMOUSE.COMプログラムが実行されて,マウス・ドライバーがシステム・ユニットの記憶域に読み込まれます。

MOUSEというコマンドの入力は、DOSプロンプトが表示されているときであればいつでも可能で、DOS始動時の直後である必要はありません。ただし、マウス・ドライバーを使用する前でなければならないのはもちろんのことです。

記憶域に読み込まれたこのプログラムは,　DOSの再始動(システム・リセットなど）によって記憶域から消去されてしまいますので,その場合には上記のステップに従って再度読み込むことが必要です。

いちいちコマンドを入力するのがめんどうな場合は、AUTOEXEC.BATファイルを使って、DOS始動時に、自動的にマウス・ドライバーを読み込むようにすることができます。

AUTOEXEC.BATファイル(なければ作成してください)に、次の1行を加えてください。

```
MOUSE
```

コマンドを入力してマウス・ドライバーを読み込む方法を使う場合は、システムを始動する度に、上記の操作を繰り返す必要があります。

第2章 マウス・インターフェース

この章では、マウスを使ったアプリケーション・プログラムを開発するために必要なインターフェースに関する次の事柄を説明します。

- 画面モード
- 画面の座標
- カーソル
- ボタン
- マウスの移動距離の単位

画面モード

マウス・ドライバーは、DOS/Vがサポートする以下の画面モードで動きます。

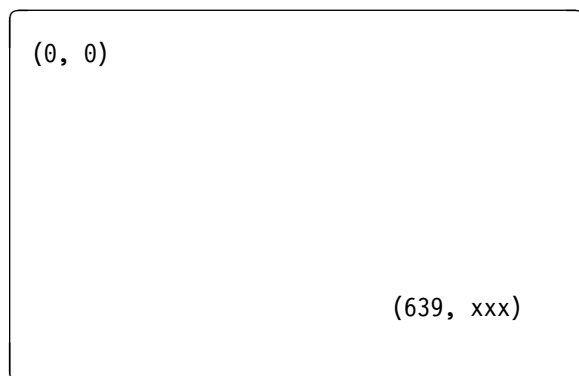
画面 モード番号	カラー	モード	仮想画面	日本語 モード	英語 モード
0	16	テキスト	640 × 200		○
1	16	テキスト	640 × 200		○
2	16	テキスト	640 × 200		○
3	16	テキスト	640 × 200	○	○
4	4	グラフィック	640 × 200		○
5	4	グラフィック	640 × 200		○
6	2	グラフィック	640 × 200		○
7	モノクロ	テキスト	640 × 200		○
D	16	グラフィック	640 × 200		○
E	16	グラフィック	640 × 200		○
F	モノクロ	グラフィック	640 × 350		○
10	16	グラフィック	640 × 350		○
11	2	グラフィック	640 × 480	○	○
12	16	グラフィック	640 × 480	○	○
13	256	グラフィック	640 × 200		○
72	16	グラフィック	640 × 480	○	
73	16	テキスト	640 × 200	○	

仮想画面に関しては、次項の説明を参照してください。

画面の座標

マウス・ドライバーは、画面上の画素や文字の位置を、仮想画面座標を用いて表します。仮想画面座標は、画面上の左上端を原点(0,　0)とし、水平方向にx軸を、垂直方向にy軸をとります（右方向がx軸の正、下方向がy軸の正とします）。

画面モードによらず、xの最小値は0、最大値は639です。yの値は最小値が0、最大値は画面モードにより異なります。yの値の最大値は「画面モード」の項を参照してください。

```
(0, 0)



                    (639, xxx)
```

xxx：199, 349, 479（画面モードにより異なる）

マウス・ドライバーの各機能を使って画素や文字を操作するときは、正しい座標を与えるようにしてください。範囲外の座標が与えると、マウス・ドライバーはその値を範囲内の値に変更して処理を行いますので、意図した図形が正しく表示されないことになります。

カーソル

DOS/Vのマウス・ドライバーは、画面モードによってグラフィック・カーソルとテキスト・カーソルの2種類のカーソルを用意しています。

グラフィック・カーソル

グラフィック・カーソルは、画面モードがグラフィック・モードのときに表示されます。グラフィック・カーソルは、たとえば矢印の形をしたもので、画面上の1点を指すために使用します。また、カーソルの形によって、現在の状況や、次に行う動作を表現することもできます。

カーソルの形は、16 × 16ドットで、0と1で表したANDマスクとXORマスクの2つのマスクを使って定義します。ANDマスクとXORマスクは、それぞれ32バイトの情報であ

るため、1つのカーソルを定義するのに64バイトの領域が必要となります。マウス・ドライバーには、あらかじめ標準のカーソルが用意されています。標準カーソルのANDマスクとXORマスクは次のとおりです。

```
ANDマスク                XORマスク

0011111111111111         0000000000000000
0001111111111111         0100000000000000
0000111111111111         0110000000000000
0000011111111111         0111000000000000
0000001111111111         0111100000000000
0000000111111111         0111110000000000
0000000011111111         0111111000000000
0000000001111111         0111111100000000
0000000000111111         0111100000000000
0000000111111111         0110110000000000
0000000111111111         0100110000000000
0011000011111111         0000011000000000
1111000011111111         0000011000000000
1111100001111111         0000001100000000
1111100001111111         0000001100000000
1111100001111111         0000000000000000
```

カーソルは画面上の１点を示すために使用しますから、カーソルの中のどの点でその１点を指すのかを決めておく必要があります。この点を「カーソルの指示点」といいます。標準カーソルの場合は、矢印の先端に指示点があります。

標準カーソルの他に、機能9を使っていろいろな形のカーソルを定義することができます。カーソルを定義するときにカーソルの指示点も指定します。付録Aに、いくつかのカーソルを定義した例を挙げています。

ANDマスクとXORマスク

カーソルは、カーソルを置く16 × 16ドットの範囲の画面に対して、ANDマスクとのAND演算を行い、その結果に対してXORマスクをXOR演算した結果として表示されます。

演算の詳細は次のとおりです。

まず、画面の対応するドットごとにANDマスクとAND演算を行います。次に、AND演算の結果とXORマスクでXOR演算を行います。AND演算、XOR演算は次の形のものです。

A	B	A AND B
0	0	0
1	0	0
0	1	0
1	1	1

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

元の画面のビットの状態、0か1（下表では0/1と表示しています）にAND演算、XOR演算を行った結果は、次のようになります。

元のビット	ANDマスク	XORマスク	結果のビット	結果の画面の状態
0/1	0	0	0/0	0
0/1	0	1	1/1	1
0/1	1	0	0/1	元の画面のビットのまま
0/1	1	1	1/0	元の画面のビットを反転

標準カーソルを例にとると、演算の結果、次のようになります。

```
0 0 ␣ ␣ ␣ ␣ ␣ ␣ ␣ ␣ ␣ ␣ ␣ ␣ ␣ ␣
0 1 0 ␣ ␣ ␣ ␣ ␣ ␣ ␣ ␣ ␣ ␣ ␣ ␣ ␣
0 1 1 0 ␣ ␣ ␣ ␣ ␣ ␣ ␣ ␣ ␣ ␣ ␣ ␣
0 1 1 1 0 ␣ ␣ ␣ ␣ ␣ ␣ ␣ ␣ ␣ ␣ ␣
0 1 1 1 1 0 ␣ ␣ ␣ ␣ ␣ ␣ ␣ ␣ ␣ ␣
0 1 1 1 1 1 0 ␣ ␣ ␣ ␣ ␣ ␣ ␣ ␣ ␣
0 1 1 1 1 1 1 0 ␣ ␣ ␣ ␣ ␣ ␣ ␣ ␣
0 1 1 1 1 1 1 1 0 ␣ ␣ ␣ ␣ ␣ ␣ ␣
0 1 1 1 1 0 0 0 0 0 ␣ ␣ ␣ ␣ ␣ ␣
0 1 1 0 1 1 0 ␣ ␣ ␣ ␣ ␣ ␣ ␣ ␣ ␣
0 1 0 0 1 1 0 ␣ ␣ ␣ ␣ ␣ ␣ ␣ ␣ ␣
0 0 ␣ ␣ 0 1 1 0 ␣ ␣ ␣ ␣ ␣ ␣ ␣ ␣
␣ ␣ ␣ ␣ 0 1 1 0 ␣ ␣ ␣ ␣ ␣ ␣ ␣ ␣
␣ ␣ ␣ ␣ ␣ 0 1 1 0 ␣ ␣ ␣ ␣ ␣ ␣ ␣
␣ ␣ ␣ ␣ ␣ 0 1 1 0 ␣ ␣ ␣ ␣ ␣ ␣ ␣
␣ ␣ ␣ ␣ ␣ 0 0 0 0 ␣ ␣ ␣ ␣ ␣ ␣ ␣
```

␣　元の画面のビットのまま

テキスト・カーソル

テキスト・カーソルは画面モードがテキスト・モードのときに表示されます。テキスト・カーソルは画面上の1バイト文字分を1単位として移動します。

テキスト・カーソルには、ソフトウェア・テキスト・カーソルとハードウェア・テキスト・カーソルの2種類あります。ただし、日本語モードではハードウェア・テキスト・カーソルはサポートされません。

ソフトウェア・テキスト・カーソル

ソフトウェア・テキストカーソルの定義には、グラフィック・カーソルと同じくANDマスクとXORマスクを用います。ただし、ANDマスクとXORマスクは、それぞれ16ビット（2バイト）の情報です。

カーソルは、カーソルを置く1文字分情報（文字コードと属性）に対してANDマスクとのAND演算を行い、その結果に対してXORマスクをXOR演算した結果として表示されます。AND,　XOR演算の詳細は、グラフィック・カーソルの項を参照してください。

マウス・ドライバーに用意されている標準のソフトウェア・テキスト・カーソルを例にとると、次のようになります。

ビット	文字情報の意味
0 - 7	文字コード（ASCIIコード）
8 - 10	前景色
11	強調
12 - 14	背景色
15	明滅 （日本語モードではサポートしません）

ビット	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
元のビット	0/1	0/1			0/1	0/1			0/1							
ANDマスク (標準の値)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
XORマスク (標準の値)	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	0
結果のビット	0/1	1/0			0/1	1/0			0/1							
結果の 画面の状態	前・背景色のビットがそれぞれ反転								文字コード変わらず							

ハードウェア・テキスト・カーソル

英語モードのテキスト・モードでのみ表示されます。ハードウェア・テキスト・カーソルはDOSのプロンプトに表示されるカーソルと同じ種類のもので、幅が1文字分、高さが可変の明滅する走査線からなります。高さを決定するのに走査線開始行と終了行を設定して、機能10を呼び出します。

ボタン

マウス・ボタンの状態には、ボタンが下がっている「押した」状態と、ボタンが上がっている「離した」状態の2つがあります。

マウス・ドライバーはマウス・ボタンが押されているか、離されているかの状態を読み取ります。また、押したり離したりした回数も数えています。カウントはプログラム内部のカウンターで行い、対応するボタンが押されたり離されたりするたびに、カウンターの数を増していきます。この値は機能0が呼び出されたとき、および機能5または6でその値が読み取られたときに、0にクリアされます。

マウスの移動距離の単位

机の上でマウスを動かすと、マウス・アダプターは底面のボールの動きからマウスが移動した方向と距離を検知し、その情報を数値に変換してマウス・ドライバーに伝えます。このとき、どれだけ動いたかという数値の単位はおよそ0.5mmを1としたもので、これを「**マウス△**」と呼びます。

マウス・ドライバーは、この数値をもとに画面上のカーソルを動かしますが、必ずしも1マウス△で1ドット分カーソルが動くのではなく、自由に設定することができます。このマウスの移動距離とカーソルの移動距離の割合を**移動比率**と呼びます。移動比率は、x方向、y方向をそれぞれ機能15で設定します。

第3章 マウス機能を呼び出すには

マクロ・アセンブラーで書かれたプログラムからマウス機能を呼び出すには、次のようにします。

1. マウス機能の機能番号をAXレジスターに、また必要なパラメーター値をBX, CX, DXの各レジスターに設定してください。
2. INT 33H命令を実行してください。

例

次の命令はカーソルを表示した後、指定した座標に位置づけます。

```
;
; カーソルの表示
;
          MOV AX, 1H            ;機能番号(1)の設定
          INT 33H               ;マウス機能の呼び出し
;
; カーソルの位置づけ
;
          MOV AX, 4H            ;機能番号(4)の設定
          MOV CX, 0H            ;X座標値の設定
          MOV DX, 0H            ;Y座標値の設定
          INT 33H               ;マウス機能の呼び出し
```

第4章　マウス機能

マウス・ドライバーには、次の機能が用意されています。

機能番号	機能の概要
0	マウス機能を初期設定する
1	カーソルを表示する
2	カーソルを消去する
3	カーソル位置とボタンの状態を読み取る
4	カーソルを位置付ける
5	ボタンが押された回数と位置を読み取る
6	ボタンが離された回数と位置を読み取る
7	カーソルのx方向の動きの範囲を定める
8	カーソルのy方向の動きの範囲を定める
9	グラフィック・カーソルの形状を定める
10	テキスト・カーソルを設定する
11	マウスが動いた距離を読み取る
12	割り込みサブルーチンを定める
13	ライト・ペン・エミュレーションを可能にする
14	ライト・ペン・エミュレーションを不可にする
15	マウスの移動比率を定める
16	カーソルを消去する画面上の範囲を指定する
20	ユーザー割り込みサブルーチンを交換する
21	マウス・ドライバーの状態の保管に必要な記憶域の大きさを得る
22	マウス・ドライバーの状態を保管する
23	マウス・ドライバーの状態を回復する
24	代替割り込みサブルーチンを設定する
25	代替割り込みサブルーチンのアドレスを獲得する
26	マウス感度を設定する

機能番号	機能の概要
27	マウス感度を獲得する
29	CRTページ番号を設定する
30	CRTページ番号を獲得する
31	マウス・ドライバーを使用不可にする
32	マウス・ドライバーを使用可にする
33	ソフトウェア・リセットをする

注: マウスの使用を開始するには、最初に「マウスの初期設定（機能番号0）」を実行する必要があります。また、終了する場合にも「マウスの初期設定」を実行する必要があります。

この章では、各機能を機能番号順に説明します。

次ページ以降の説明で、マウス機能の呼び出し時のパラメーターは、**パラメーター**の表にまとめています。

また、パラメーターの値を、ビット0=1,　ビット1=1,・・・と表現していることがあります(機能3，5，6，12など)。これは、このビットがオンである場合を表しており、2進数で表現したときに、対応する桁(ビット0が1桁目、ビット1が2桁目、・・・になります）が1になっていることを意味します。

ビット0=1：左ボタンが押されている
ビット1=1：右ボタンが押されている

これは、ビット0(2進数の1桁目)が1であれば、左ボタンが押されていることを表し、0であれば離されていることを表します。同様に、ビット1（2進数の2桁目）が1であれば、右ボタンが押されていること表し、0であれば離されていることを表します。

次表に各ビットがオンである場合の10進数の値を示します。

	10進数による重み
ビット0=1	1（2^0)
ビット1=1	2（2^1)
ビット2=1	4（2^2)
ビット3=1	8（2^3)
ビット4=1	16（2^4)

たとえば、ビット0とビット3が1の場合は、「10進数による重み」の対応する値を加算して、1 + 8で、9ということになります。

マウスの初期設定－機能0

目的

マウスが使用できるかどうかの情報を返し、マウス機能を初期状態に設定します。

パラメーター

	入力	出力
AX	機能番号、0	マウスが使用できるかどうか =0 マウス使用不可 =-1 マウス使用可
BX		マウス・ボタンの数
CX		
DX		

説明

マウス機能の初期状態は次のとおりです。

項目		**状態**
カーソルの表示		表示しない
カーソルの位置		画面の中央
カーソル形	テキスト・カーソル	反転表示
	グラフィック・カーソル	標準の矢印
カーソルの指示点（グラフィック・カーソル）		(0, 0)
カーソルの移動範囲	x方向	0～639
	y方向	画面モードによる
マウスの移動比率	x方向のマウス&ys79.の数:仮想画面座標	8:8
	y方向のマウス&ys79.の数:仮想画面座標	16:8
割り込みのためのマスク		すべて0

マウスの使用を開始するには、この「マウスの初期設定」を行う必要があります。また、マウスの使用を終了する場合にも、「マウスの初期設定」を行う必要があります。

カーソルの表示－機能1

目的

カーソルを画面に表示します。

パラメーター

	入力	出力
AX	機能番号、1	
BX		
CX		
DX		

説明

マウス・ドライバーは、カーソル表示のためのフラグをプログラム内部に保持しています。カーソルはこのフラグが0のときだけ表示されます。フラグの初期値は-1で、機能1を実行するたびに1だけ増やされます。ただし、すでにフラグの値が0のときは、変化しません。したがって、フラグの値が-1より小さい場合に機能1を実行してもカーソルは表示されません。一方、機能2（カーソルの消去;次項参照）を実行すると、フラグは1だけ減らされます（フラグの値にかかわらず常に1だけ減らされます）。

カーソルの消去－機能2

目的

カーソルを画面から消去します。

パラメーター

	入力	出力
AX	機能番号、2	
BX		
CX		
DX		

説明

カーソルを画面に表示しないようにします。このとき、カーソル表示のためのフラグ（機能1の説明参照）を1だけ減らします。カーソルの位置は、画面に表示されていなくても、マウスの動きに従って変わります。カーソルが表示されている位置に文字や図形を表示する場合は、カーソルを一時消去する必要があります。

カーソル位置とボタンの状態の読み取り－機能3

目的

カーソルの位置と、ボタンが押されているか離されているかの情報を読み取ります。

パラメーター

	入力	出力
AX	機能番号、3	
BX		ボタンの状態 ビット0=1 左ボタンが押されている ビット1=1 右ボタンが押されている
CX		カーソルのx座標
DX		カーソルのy座標

カーソルの位置付け－機能4

目的

カーソルを、指定した位置に置きます。

パラメーター

	入力	出力
AX	機能番号、4	
BX		
CX	カーソルのx座標	
DX	カーソルのy座標	

説明

カーソルの位置がカーソルの移動範囲を超えた場合、マウス・ドライバーは、カーソル位置を境界上に設置します。

ボタンが押された回数と位置の読み取り－機能5

目的

指定したボタンが押された回数と、指定したボタンが最後に押されたときのカーソル位置および現在のボタンの状態を読み取ります。

パラメーター

	入力	出力
AX	機能番号、5	現在のボタンの状態 ビット0=1 左ボタンが押されている ビット1=1 右ボタンが押されている
BX	ボタンの指定 =0 左ボタン =1 右ボタン	指定したボタンが押された回数
CX		最後に、指定したボタンが押されたときのカーソルのx座標
DX		最後に、指定したボタンが押されたときのカーソルのy座標

説明

CXおよび**DX**レジスターの値は、最後にボタンが押されたときのカーソルの位置です。現在のカーソル位置ではありません。

機能5が呼び出されると、**BX**レジスターで指定したボタンが押された回数は、0にクリアされます。

ボタンが離された回数と位置の読み取り－機能6

目的

指定したボタンが離された回数と、指定したボタンが最後に離されたときのカーソル位置および現在のボタンの状態を読み取ります。

パラメーター

	入力	出力
AX	機能番号、6	現在のボタンの状態 ビット0=1 左ボタンが押されている ビット1=1 右ボタンが押されている
BX	ボタンの指定 =0 左ボタン =1 右ボタン	指定したボタンが離された回数
CX		最後に、指定したボタンが離されたときのカーソルのx座標
DX		最後に、指定したボタンが離されたときのカーソルのy座標

説明

CXおよび**DX**レジスターの値は、最後にボタンが離されたときのカーソルの位置です。現在のカーソル位置ではありません。

機能6が呼び出されると、**BX**レジスターで指定したボタンが離された回数は、0にクリアされます。

カーソルの移動範囲（x方向）の指定－機能7

目的

画面上でカーソルが動く範囲を定めます。機能7では、x方向の最小と最大を定めます。

パラメーター

	入力	出力
AX	機能番号、7	
BX		
CX	カーソルの移動範囲の左端の座標	
DX	カーソルの移動範囲の右端の座標	

説明

この機能を呼び出した後は、カーソルの動きはこの範囲内に限られます。呼び出しの時点で、カーソルが範囲外にあれば、範囲内の境界上に置かれます。**CX**レジスターの値が**DX**レジスターの値よりも大きいときには、2つの値は入れ換えられます。

カーソルの移動範囲（y方向）の指定－機能8

目的

画面上でカーソルが動く範囲を定めます。機能8では、y方向の最小と最大を定めます。

パラメーター

	入力	出力
AX	機能番号、8	
BX		
CX	カーソルの移動範囲の上端の座標	
DX	カーソルの移動範囲の下端の座標	

説明

この機能を呼び出した後は、カーソルの動きはこの範囲内に限られます。呼び出しの時点で、カーソルが範囲外にあれば、範囲内の境界上に置かれます。**CX**レジスターの値が**DX**レジスターの値よりも大きいときには、2つの値は入れ換えられます。

グラフィック・カーソルの形の定義－機能9

目的

グラフィック・カーソルの形と指示点を定義します。

パラメーター

	入力	出力
AX	機能番号、9	
BX	カーソルの指示点のx座標(－128˜127)	
CX	カーソルの指示点のy座標(－128˜127)	
ES:DX	カーソルの形が入っている領域の先頭アドレス	

説明

グラフィック・カーソルの形は、64バイトの大きさの、0と1からなるANDマスクとXORマスクで定義します。ANDマスクとXORマスクの値を隣り合った2つの領域に割り当て、その先頭のアドレスを**DX**レジスターにセットします。呼び出す前に、この領域のセグメント・アドレスが**ES**レジスターにセットされていなければなりません。

指示点を与える座標の原点(0，0)は、カーソル(16 ×16の大きさ）の左上隅です。指示点はx方向、y方向とも－128˜127の範囲で指定できますが、通常0˜15の値を用います。

機能0でマウス・ドライバーの標準の矢印カーソルが設定されます。

例

標準の矢印カーソルの形の定義を以下に示します。

```
;
;  矢印カーソルのANDマスクの定義、指示点(0, 0)
;
CURSOR_MASK    DW     03FFFH      ;0011111111111111
               DW     01FFFH      ;0001111111111111
               DW     00FFFH      ;0000111111111111
               DW     007FFH      ;0000011111111111
               DW     003FFH      ;0000001111111111
               DW     001FFH      ;0000000111111111
               DW     000FFH      ;0000000011111111
               DW     0007FH      ;0000000001111111
               DW     0003FH      ;0000000000111111
               DW     0001FH      ;0000000000011111
               DW     001FFH      ;0000000111111111
               DW     010FFH      ;0001000011111111
               DW     030FFH      ;0011000011111111
               DW     0F87FH      ;1111100001111111
               DW     0F87FH      ;1111100001111111
               DW     0FC7FH      ;1111110001111111
;
;  矢印カーソルのXORマスク定義、指示点(0, 0)
;
               DW     00000H      ;0000000000000000
               DW     04000H      ;0100000000000000
               DW     06000H      ;0110000000000000
               DW     07000H      ;0111000000000000
               DW     07800H      ;0111100000000000
               DW     07C00H      ;0111110000000000
               DW     07E00H      ;0111111000000000
               DW     07F00H      ;0111111100000000
               DW     07F80H      ;0111111110000000
               DW     07C00H      ;0111110000000000
               DW     06C00H      ;0110110000000000
               DW     04600H      ;0100011000000000
               DW     00600H      ;0000011000000000
               DW     00300H      ;0000001100000000
               DW     00300H      ;0000001100000000
               DW     00000H      ;0000000000000000
```

テキスト・カーソルの設定－機能10

目的

テキスト・カーソルの種類およびその形を定義します。

パラメーター

	入力	出力
AX	機能番号、10	
BX	カーソルの選択(0,1) 0=ソフトウェア・カーソル 1=ハードウェア・カーソル	
CX	ANDマスクの値、または走査線開始行	
DX	XORマスクの値、または走査線終了行	

説明

テキスト・カーソルの種類（ソフトウェア・テキスト・カーソルかハードウェア・テキスト・カーソル）とその形を定義します。

マウスの移動距離の読み取り－機能11

目的

最後に機能11が呼び出されたときのマウスの位置と、現在の位置の相対距離を読み取ります。

パラメーター

	入力	出力
AX	機能番号、11	
BX		
CX		x方向のマウス△の数
DX		y方向のマウス△の数

説明

最後に機能11が呼び出されたときのマウスの位置から、現在の位置までの相対距離をおよそ0.5mmを1単位として返します。返される値は、両方向とも–32768〜32767の範囲です。

相対位置は、画面の右方向と下方向をそれぞれx方向の+方向、y方向の+方向で表します。

割り込みサブルーチンの設定－機能12

目的

アプリケーション・プログラム内のサブルーチンを呼び出すための割り込み状態の設定と、呼び出すサブルーチンの設定を行います。

パラメーター

	入力	出力
AX	機能番号、12	
BX		
CX	割り込みマスク	
ES:DX	割り込みアドレス	

説明

CXレジスターには、マウスがどのように操作されたときに、マウス・ドライバーが、指定したサブルーチンに制御を移すかを設定します。

ビット	呼び出し状態	10進数による重み
ビット0 =1	カーソル位置が変更された	1
ビット1 =1	左ボタンが押された	2
ビット2 =1	左ボタンが離された	4
ビット3 =1	右ボタンが押された	8
ビット4 =1	右ボタンが離された	16

サブルーチンを呼び出すには、対応するビットを1に設定した割り込みマスクの値を**CX**レジスターに設定し、サブルーチンの先頭アドレスを**DX**レジスターに設定します。呼び出す前に、このサブルーチンのセグメント・アドレスが**ES**レジスターにセットされていなければなりません。マウス・ドライバーは、**CX**レジスターで設定したいくつかの状態のうちどれか1つでも当てはまる状態になれば、実行中のアプリケーショ

ン・プログラムの実行を中断し、設定されたサブルーチンを実行します。このサブルーチンの実行が終わると、中断したアプリケーション・プログラムの実行を再開します。

アプリケーションを終了する際に、以降、サブルーチンが呼び出されないようにする必要があります。機能0を実行すると、自動的に呼び出しの解除が行われます。

指定したサブルーチンに制御が移るとき、次の情報がレジスターに読み込まれています。

レジスター	情報
AX	マウスの状態 (10進数による重み) ビット0 =1：カーソル位置が変更された (1) ビット1 =1：左ボタンが押された (2) ビット2 =1：左ボタンが離された (4) ビット3 =1：右ボタンが押された (8) ビット4 =1：右ボタンが離された (16)
BX	ボタンの状態 ビット0 =1：左ボタンが押されている ビット1 =1：右ボタンが押されている
CX	カーソルのx座標
DX	カーソルのy座標
SI	x方向のマウスΔの数
DI	y方向のマウスΔの数

注: DSレジスターはマウス・ドライバー内を指しています。サブルーチンは必要であれば、DSレジスターを適当な値に設定しなければなりません。

ライト・ペン・エミュレーション・モード　オン－機能13

目的

ライト・ペン・エミュレーションを可能にします。

パラメーター

	入力	出力
AX	機能番号、13	
BX		
CX		
DX		

説明

ライト・ペン・エミュレーション・モードをオンにすると、マウスはライト・ペンの働きをするようになります。カーソルの位置はライト・ペンの位置を示し、マウスの両方のボタンを押すとライト・ペンを押し下げたことになります。

ライト・ペン・エミュレーション・モード　オフ－機能14

目的

ライト・ペン・エミュレーションを不可にします。

パラメーター

	入力	出力
AX	機能番号、14	
BX		
CX		
DX		

マウスの移動比率の設定－機能15

目的

マウスの動きが、画面上のカーソルの動きに伝えられる距離の割合を指定します。

パラメーター

	入力	出力
AX	機能番号、15	
BX		
CX	x方向のマウスΔの数(1～32767)	
DX	y方向のマウスΔの数(1～32767)	

説明

機能15は、画面上のカーソルを仮想画面座標で8だけ動かすのに必要なマウスの移動距離を指定するものです。この距離は、約0.5mmを1単位とするマウスΔの数で指定します。この値は、1～32767の範囲になければなりません。

機能15を呼び出さなかった場合、移動距離のマウスΔの数は、x方向は8、y方向は16です。

カーソル消去範囲の指定－機能16

目的

カーソルを消去する画面上の範囲を指定します。

パラメーター

	入力	出力
AX	機能番号、16	
BX		
CX	消去範囲の左上のx座標	
DX	消去範囲の左上のy座標	
SI	消去範囲の右下のx座標	
DI	消去範囲の右下のy座標	

説明

機能16は、アプリケーション・プログラムが画面上に文字や図形を表示する場合に呼び出します。文字や図形を表示する領域をカーソルの消去範囲に指定して機能16を呼び出すと、この領域にはカーソルは表示されません。文字や図形の表示が終わったあとで、機能1(カーソルの表示)を呼び出して、消去範囲を解除します。機能16は、機能2(カーソルの消去)と同じような働きをしますが、処理にかかる時間は短かくて済みます。

割り込みサブルーチンの交換－機能20

目的

アプリケーション・プログラム内のサブルーチンを呼び出すための割り込み状態の設定と呼び出すサブルーチンの設定を新たに行います。元の割り込みマスクと割り込みアドレスが返されます。

パラメーター

	入力	出力
AX	機能番号、20	
BX		
CX	新しい割り込みマスク	元の割り込みマスク
ES:DX	新しい割り込みアドレス	元の割り込みアドレス

説明

CXレジスターには、マウスがどのように操作されたときに、マウス・ドライバーが、指定したサブルーチンに制御を移すかを設定します。

ビット	呼び出し状態	10進数による重み
ビット0 =1	カーソル位置が変更された	1
ビット1 =1	左ボタンが押された	2
ビット2 =1	左ボタンが離された	4
ビット3 =1	右ボタンが押された	8
ビット4 =1	右ボタンが離された	16

サブルーチンを呼び出すには、対応するビットを1に設定した割り込みマスクの値を**CX**レジスターに設定し、サブルーチンの先頭アドレスを**DX**レジスターに設定します。呼び出す前に、このサブルーチンのセグメント・アドレスが**ES**レジスターにセットされていなければなりません。マウス・ドライバーは、**CX**レジスターで設定したいくつかの状態のうちどれか1つでも当てはまる状態になれば、実行中のアプリケーション・プログラムの実行を中断し、設定されたサブルーチンを実行します。このサブルーチンの実行が終わると、中断したアプリケーション・プログラムの実行を再開します。

アプリケーションを終了する際に、以降、サブルーチンが呼び出されないようにする必要があります。機能0を実行すると、自動的に呼出しの解除が行われます。

元の割り込みアドレスは、セグメントがESレジスターに、オフセットがDXレジスターに返されます。

指定したサブルーチンに制御が移るとき、次の情報がレジスターに読み込まれています。

レジスター	情報
AX	マウスの状態 (10進数による重み) ビット0 =1：カーソル位置が変更された (1) ビット1 =1：左ボタンが押された (2) ビット2 =1：左ボタンが離された (4) ビット3 =1：右ボタンが押された (8) ビット4 =1：右ボタンが離された (16)
BX	ボタンの状態 ビット0 =1：左ボタンが押されている ビット1 =1：右ボタンが押されている
CX	カーソルのx座標
DX	カーソルのy座標
SI	x方向のマウスΔの数
DI	y方向のマウスΔの数

注: DSレジスターはマウス・ドライバー内を指しています。サブルーチンは必要であれば、DSレジスターを適当な値に設定しなければなりません。

マウス・ドライバーの状態の保管に必要な記憶域の大きさを得る－機能21

目的

現在のマウス・ドライバーの状態を保管するのに必要なバッファーの大きさを返します。

パラメーター

	入力	出力
AX	機能番号、21	
BX		機能22, 23で必要なバッファーの大きさ
CX		
DX		

説明

マウスを使用している別のプログラムへ一時的に割り込みの制御を移す場合に機能22や機能23と共に使用します。

マウス・ドライバーの状態の保管－機能22

目的

割り当てられたバッファーにマウス・ドライバーの状態を保管します。

パラメーター

	入力	出力
AX	機能番号、22	
BX		
CX		
ES:DX	バッファーの先頭アドレス	

説明

マウスを使用している別のプログラムへ一時的に割り込みの制御を移す場合に機能21や機能23と共に使用します。

マウス・ドライバーの状態の保管に必要なバッファーの大きさを機能21で調べて、記憶域を割り当てた後で、機能22を呼び出してください。

呼び出す前に、この保管バッファーのセグメント・アドレスが**ES**レジスターにセットされていなければなりません。

マウス・ドライバーの状態の回復－機能23

目的

機能22で保管したマウス・ドライバーの状態を回復します。

パラメーター

	入力	出力
AX	機能番号、23	
BX		
CX		
ES:DX	バッファーの先頭アドレス	

説明

マウスを使用している別のプログラムへ一時的に割り込みの制御を移す場合に機能21や機能22と共に使用します。

機能22で保管したマウス・ドライバーの状態を回復するために、割り込みプログラムの終わりで機能23を呼び出します。呼び出す前に、機能22で指定した保管バッファーのセグメント・アドレスが**ES**レジスターにセットされていなければなりません。

代替割り込みサブルーチンの設定－機能24

目的

アプリケーション・プログラム内のサブルーチンを呼び出すための割り込み状態の設定と、呼び出すサブルーチンの設定を行います。この機能は、割り込みマスクにキー・ストロークの組み合わせが含まれること以外は機能12と同じです。この機能を呼び出すと、最大3つのルーチンまで定義することができます。

パラメーター

	入力	出力
AX	機能番号、24	=負数:エラー
BX		
CX	割り込みマスク	
ES:DX	割り込みアドレス	

説明

CXレジスターには、マウスおよびがどのように操作されたときに、マウス・ドライバーが、指定したサブルーチンに制御を移すかを設定します。

ビット	呼び出し状態	10進数による重み
ビット0 =1	カーソル位置が変更された	1
ビット1 =1	左ボタンが押された	2
ビット2 =1	左ボタンが離された	4
ビット3 =1	右ボタンが押された	8
ビット4 =1	右ボタンが離された	16
ビット5 =1	ボタンの操作と共にシフト・キーが押された	32
ビット6 =1	ボタンの操作と共にCtrlキーが押された	64
ビット7 =1	ボタンの操作と共に前面キーが押された	128

サブルーチンを呼び出すには、対応するビットを1に設定した割り込みマスクの値を**CX**レジスターに設定し、サブルーチンの先頭アドレスを**DX**レジスターに設定します。呼び出す前に、このサブルーチンのセグメント・アドレスが**ES**レジスターにセットされていなければなりません。マウス・ドライバーは、**CX**レジスターで設定したいくつかの状態のうちどれか1つでも当てはまる状態になれば、実行中のアプリケーション・プログラムの実行を中断し、設定されたサブルーチンを実行します。このサブルーチンの実行が終わると、中断したアプリケーション・プログラムの実行を再開します。

指定したサブルーチンに制御が移るとき、次の情報がレジスターに読み込まれています。

レジスター	情報
AX	マウスの状態 (10進数による重み) ビット0 =1：カーソル位置が変更された (1) ビット1 =1：左ボタンが押された (2) ビット2 =1：左ボタンが離された (4) ビット3 =1：右ボタンが押された (8) ビット4 =1：右ボタンが離された (16)
BX	ボタンの状態 ビット0 =1：左ボタンが押されている ビット1 =1：右ボタンが押されている
CX	カーソルのx座標
DX	カーソルのy座標
SI	x方向のマウスΔの数
DI	y方向のマウスΔの数

注: DSレジスターはマウス・ドライバー内を指しています。サブルーチンは必要であれば、DSレジスターを適当な値に設定しなければなりません。

代替割り込みサブルーチンのアドレス獲得－機能25

目的

指定した割り込みマスクに対応する、現在の代替割り込みサブルーチンのアドレスを獲得します。

パラメーター

	入力	出力
AX	機能番号、25	エラーの状態
BX		割り込みアドレスのセグメント
CX	割り込みマスク	割り込みマスク
DX		割り込みアドレスのオフセット

説明

機能24で代替割り込みサブルーチンを設定する前に、現在指定されている割り込みサブルーチンのアドレスを獲得します。アプリケーション・プログラムは、機能24で設定した割り込みサブルーチンの呼び出しが不必要になると、機能25で獲得したアドレスを用いて、元の値に再設定しなければなりません。入力としてCXレジスターに設定した割り込みマスクに対応する、現在の割り込みアドレスの値が返されます。対応する割り込みサブルーチンが以前に設定されていない場合、AXレジスターに–1が返されます。

マウス感度の設定－機能26

目的

マウスの移動比率を調整するための係数を設定します。

パラメーター

	入力	出力
AX	機能番号、26	
BX	x方向のマウス移動比率の係数(1˜100)	
CX	y方向のマウス移動比率の係数(1˜100)	

説明

マウスの移動比率および倍速境界値そのものを設定するわけではありません(機能15を使用してください)。マウス・ドライバーは、機能26で設定された係数を用いてマウスの移動比率に対して演算を行い、演算後の値を実際のマウスの移動比率として用います。係数が50のとき、演算後の値も小さく、50より大きいときは演算後の値も大きくなります。機能26を呼び出さないときの係数の値は50です。また、係数を100より大きくして機能26を呼び出しても、100に設定されます。

マウス感度の獲得－機能27

目的

機能26で設定された値を返します。

パラメーター

	入力	出力
AX	機能番号、27	
BX		x方向のマウス移動比率の係数(1˜100)
CX		y方向のマウス移動比率の係数(1˜100)

説明

現在のマウス移動比率の係数を獲得します。

CRTページ番号の設定－機能29

目的

マウス・カーソルを表示するCRTページを指定します。

パラメーター

	入力	出力
AX	機能番号、29	
BX	カーソルを表示するCRTページ	
CX		
DX		

説明

CRTページに関しては「DOS/V BIOSインターフェース 技術解説編」を参照してください。

CRTページ番号の獲得－機能30

目的

マウス・カーソルが表示されるCRTページを返します。

パラメーター

	入力	出力
AX	機能番号、30	
BX		カーソルが表示されるCRTページ
CX		
DX		

説明

CRTページに関しては「DOS/V BIOSインターフェース 技術解説編」を参照してください。

マウス・ドライバーの使用不可－機能31

目的

この機能は、割り込み33H (INT 33H)を除く、マウス・ドライバーが使用するすべての割り込みベクトルを、マウス・ドライバーが導入される前の値に復元します。

パラメーター

	入力	出力
AX	機能番号、31	使用不可にできなかった場合は -1
ES:BX		元のINT 33Hの割り込みベクトル
CX		
DX		

説明

ES:BXレジスターに返される値は、INT 33Hのベクトルの値をマウス・ドライバーが導入される以前の値に復元するために使用することができます。

マウス・ドライバーは、INT 10Hのベクトルと、INT 71H (8086マイクロプロセッサーの場合）またはINT 74H (i286/i386マイクロプロセッサーの場合）のベクトルを使用します。マウス・ドライバーが使用しているベクトルのうち、１つでも復元できないベクトル値があると、AXレジスターに-1が返されます。

マウス・ドライバーの使用可－機能32

目的

この機能は、マウス・ドライバーが使用するすべての割り込みベクトル値を再設定します。

パラメーター

	入力	出力
AX	機能番号、32	
BX		
CX		
DX		

説明

機能32は機能31で使用不可にしたマウス・ドライバーを再度使用可の状態にします。

ソフトウェア・リセット－機能33

目的

マウス・ドライバーの諸変数を初期化します。

パラメーター

	入力	出力
AX	機能番号、33	マウス・ドライバーが導入されている場合は -1, 導入されていない場合は 33
BX		AX = -1の場合は、マウス・ボタンの数
CX		
DX		

説明

この機能は、マウスのハードウェアを初期化しない点を除き、機能0と同じ働きをします。

付録A. グラフィック・カーソルの例

この付録には4個のグラフィック・カーソルの例を挙げています。

カーソルの形を変えることによって、マウスを使った次の動作がどのようなものかを表すことができます。たとえば、一般的に矢印は「ある項目を選択する」ときに使います。

この他にも、様々な形のカーソルをこの例のようにして定義することができます。

以降の例の注釈部分にはカーソルの形を表すビット・パターンが書かれています。

例1：四角形

カーソルの指示点は、図形の中心にあります。

```
;
;   四角形カーソルのANDマスク定義、指示点(7, 7)
;
CURSOR_MASK    DW     0FFFFH      ;1111111111111111
               DW     0FFFFH      ;1111111111111111
               DW     0FFFFH      ;1111111111111111
               DW     0E007H      ;1110000000000111
               DW     0E007H      ;1110000000000111
               DW     0E7E7H      ;1110011111100111
               DW     0E7E7H      ;1110011111100111
               DW     0E667H      ;1110011001100111
               DW     0E667H      ;1110011001100111
               DW     0E7E7H      ;1110011111100111
               DW     0E7E7H      ;1110011111100111
               DW     0E007H      ;1110000000000111
               DW     0E007H      ;1110000000000111
               DW     0FFFFH      ;1111111111111111
               DW     0FFFFH      ;1111111111111111
               DW     0FFFFH      ;1111111111111111
;
;   四角形カーソルのXORマスクの定義、指示点(7, 7)
;
               DW     00000H      ;0000000000000000
               DW     00000H      ;0000000000000000
               DW     00000H      ;0000000000000000
               DW     01FF8H      ;0001111111111000
               DW     01FF8H      ;0001111111111000
               DW     01818H      ;0001100000011000
               DW     01818H      ;0001100000011000
               DW     01998H      ;0001100110011000
               DW     01998H      ;0001100110011000
               DW     01818H      ;0001100000011000
               DW     01818H      ;0001100000011000
               DW     01FF8H      ;0001111111111000
               DW     01FF8H      ;0001111111111000
               DW     00000H      ;0000000000000000
               DW     00000H      ;0000000000000000
               DW     00000H      ;0000000000000000
```

例2：えんぴつ

この例のカーソルの指示点は、えんぴつの先端です。

```
;
;   えんぴつ形カーソルのANDマスク定義、指示点(0, 15)
;
CURSOR_MASK    DW     0FFDFH      ;1111111111011111
               DW     0FFAFH      ;1111111110101111
               DW     0FF67H      ;1111111101100111
               DW     0FEDBH      ;1111111011011011
               DW     0FDB9H      ;1111110110111001
               DW     0FB76H      ;1111101101110110
               DW     0F6EDH      ;1111011011101101
               DW     0EDDBH      ;1110110111011011
               DW     0DBB7H      ;1101101110110111
               DW     0B76FH      ;1011011101101111
               DW     0BEDFH      ;1011111011011111
               DW     0BDBFH      ;1011110110111111
               DW     0BF7FH      ;1011111101111111
               DW     0BFFFH      ;1011111011111111
               DW     001FFH      ;0000000111111111
               DW     03FFFH      ;0011111111111111
;
;   えんぴつ形カーソルのXORマスク定義、指示点(0, 15)
;
               DW     00020H      ;0000000000100000
               DW     00050H      ;0000000001010000
               DW     00098H      ;0000000010011000
               DW     00124H      ;0000000100100100
               DW     00246H      ;0000001001000110
               DW     00489H      ;0000010010001001
               DW     00912H      ;0000100100010010
               DW     01224H      ;0001001000100100
               DW     02448H      ;0010010001001000
               DW     04890H      ;0100100010010000
               DW     04120H      ;0100000100100000
               DW     04240H      ;0100001001000000
               DW     04080H      ;0100000010000000
               DW     04100H      ;0100000100000000
               DW     0FE00H      ;1111111000000000
               DW     0C000H      ;1100000000000000
```

例3：星印

カーソルの指示点は、図形の中心にあります。

```
;
;  星印カーソルのANDマスク定義、指示点(7，7)
;
CURSOR_MASK    DW     0FEFFH      ;1111111011111111
               DW     0FE7FH      ;1111111001111111
               DW     0FE7FH      ;1111111001111111
               DW     0FE7FH      ;1111111001111111
               DW     0FE7FH      ;1111111001111111
               DW     0FC3FH      ;1111110000111111
               DW     08001H      ;1000000000000001
               DW     0C003H      ;1100000000000011
               DW     0F00FH      ;1111000000001111
               DW     0F81FH      ;1111100000011111
               DW     0F81FH      ;1111100000011111
               DW     0F18FH      ;1111000110001111
               DW     0F3CFH      ;1111001111001111
               DW     0E7E7H      ;1110011111100111
               DW     0EFF7H      ;1110111111110111
               DW     0FFFFH      ;1111111111111111
;
;  星印カーソルのXORマスク定義、指示点(7，7)
;
               DW     00100H      ;0000000100000000
               DW     00180H      ;0000000110000000
               DW     00180H      ;0000000110000000
               DW     00180H      ;0000000110000000
               DW     00180H      ;0000000110000000
               DW     003C0H      ;0000001111000000
               DW     07FFEH      ;0111111111111110
               DW     03FFCH      ;0011111111111100
               DW     00FF0H      ;0000111111110000
               DW     007E0H      ;0000011111100000
               DW     007E0H      ;0000011111100000
               DW     00E70H      ;0000111001110000
               DW     00C30H      ;0000110000110000
               DW     01818H      ;0001100000011000
               DW     01008H      ;0001000000001000
               DW     00000H      ;0000000000000000
```

例4：腕時計

カーソルの指示点は、図形の中心にあります。

```
;
;  腕時計カーソルのANDマスク定義、(7，7)
;
CURSOR_MASK    DW     0FF00H      ;1111111100000000
               DW     0FE80H      ;1111111010000000
               DW     0FDC0H      ;1111110111000000
               DW     0FBE0H      ;1111101111100000
               DW     0F7E0H      ;1111011111100000
               DW     0EFD8H      ;1110111111011000
               DW     0DFBCH      ;1101111110111100
               DW     0BE7EH      ;1011111001111110
               DW     07E7DH      ;0111111001111101
               DW     03FBBH      ;0011111110111011
               DW     01FD3H      ;0001111111010011
               DW     00FE3H      ;0000111111100011
               DW     007C7H      ;0000011111000111
               DW     003BFH      ;0000001110111111
               DW     0017FH      ;0000000101111111
               DW     000FFH      ;0000000011111111
;
;  腕時計カーソルのXORマスク定義、(7，7)
;
               DW     000FFH      ;0000000011111111
               DW     0017FH      ;0000000101111111
               DW     0023FH      ;0000001000111111
               DW     0041FH      ;0000010000011111
               DW     0081FH      ;0000100000011111
               DW     01027H      ;0001000000100111
               DW     02043H      ;0010000001000011
               DW     04181H      ;0100000110000001
               DW     08182H      ;1000000110000010
               DW     0C044H      ;1100000001000100
               DW     0E02CH      ;1110000000101100
               DW     0F01CH      ;1111000000011100
               DW     0F838H      ;1111100000111000
               DW     0FC40H      ;1111110001000000
               DW     0FE80H      ;1111111010000000
               DW     0FF00H      ;1111111100000000
```

テクニカル・リファレンス

DOS/V プログラミング概説編

特記事項

『DOS/V プログラミング概説編』の2ページから7ページの表、13, 14,ページのキーボードの図は、インターナショナル・ビジネス・マシーンズ・コーポレーション(以下IBMという)の許可により、IBMの出版物“IBM DOSバージョンJ5.0/V BIOSインターフェース技術解説書”、“IBM DOSバージョンJ5.0/V技術解説書”、“IBM DOSバージョンJ5.0/Vユーザーズ・ガイド”および“OADGハードウェア・インターフェース技術解説編”の内容を記載しており、これらの著作権はIBMが所有しています。

キーボードの図、図中のAXロゴマーク及び走査コード/文字コードの表は、AX協議会の転載許諾の承認の上、AX協議会発行“AXテクニカル・リファレンスガイド-1989-”の3.3章中の内容を記載しており、この部分の著作権はAX協議会が所有しています。

キーボードの図及び走査コード/文字コードの表は、株式会社東芝の使用許諾の承認の上記載しており、この部分の著作権は株式会社東芝が所有しています。

これらの内容の無断転載・複製などの著作権に抵触する使用はできません。

本書で使用されている次の用語は、米国IBM社の商標です。

IBM
Personal Computer AT (略称: PC/AT)
Personal System/2 (略称: PS/2)
Personal System/55 (略称: PS/55)

図中のAXのロゴマークはAX協議会の商標です。

本文中のESC/Pは、セイコーエプソン株式会社の登録商標です。

本書で使用されている「J-3100」は、株式会社東芝の商標です。

まえがき

パーソナル・コンピューターの普及が目覚ましく、その市場は依然として高成長が期待されます。一方、パーソナル・コンピューターのソフトウェアの共通利用に関しては、必ずしもユーザーの要求に十分応えていないというのが現状です。その根本的な理由は、ハードウェアの仕様がメーカー間で異なっているためですが、それに加えてパーソナル・コンピューターの場合、その開発経験や利用のノウハウの点で、あまりに多岐にわたっているからでもあります。

このような状況に鑑み、ソフトウェア利用の共通基盤の確立を主要な目的の一つとして『PCオープン・アーキテクチャー推進協議会』（略称OADG）が設立されました。これにより、異なるハードウェア上で稼働する多様なアプリケーションの提供が可能となり、この結果パーソナル・コンピューターの活用度が向上することになります。

OADGが目指すソフトウェア・インターフェースの基本になるのは、OADG参加各社のハードウェア上で、共通のインターフェースを提供できるDOS/Vです。OADG DOS/VはIBM DOS/Vの仕様を基本とし、それぞれのハードウェアをサポートします。

本書は、これらのシステム上で稼働するソフトウェアである、OADGアプリケーションを開発するためのプログラミング考慮点を記述します。

参考資料

1. IBM Personal System/2 and Personal Computer BIOS Interface Technical Reference (IBM S68X-2341)

2. EPSON ESC/P* リファレンスマニュアル 第2版（セイコーエプソン(株)）

目次

割り込み

DOSのハードウェア・アクセスのための基本的なインターフェースは、割り込みです。ソフトウェアの移植性を高める上でも、可能なかぎり高位のインターフェースを使用してください。BIOSインターフェースなど、低位のインターフェースを使用する場合は、ハードウェアの違いに依存する場合がありますのでサポートするハードウェアを考慮してください。DOS/Vの基本インターフェースを高位から低位に並べると次のようになります。

1. DOSファンクション・コール (INT 21H)
2. DOS割り込み (INT 20H, 22H-3FH)
3. BIOS割り込み (INT 10H, 16H, 17H)
4. INT 10H AH=FEH, FFHによるビデオ・バッファーへの直接書き込み
5. BIOS割り込み (INT 15H)
6. BIOS割り込み (その他のINT 10H-1FH)

1.から4.はOADG DOS/Vで共通です。5.と6.は、PC/AT互換機とPS/2互換機やPS/55の間でサポートする機能が一部異なります。

要求する機能が満たされるなら、より高位のインターフェースを使用してください。通常のアプリケーションでは5.以下は使用すべきではありません。

DOS, BIOSの割り込みを取り込むデバイス・ドライバー、アプリケーションなどを作る場合、元の割り込みに制御を戻すことを忘れないでください。割り込みを受けたとき、自分の取り込んだ機能かどうか検査し、サポートしていない機能であれば、元の割り込みのアドレスをチェーンしてください。

DOSファンクション・コール (INT 21H)

現在00Hから6CHまでのファンクション・コールが定義されています。

DOS/Vで使用可能なDOSファンクション・コールを以下の表にまとめました。OADGアプリケーションとして考慮すべきものには印を付けました。

詳細は“DOS/V 技術解説編”を参照してください。

番号	機能名	互換性の考慮点
00H	プログラムの終了	
01H	コンソール入力(エコーあり)	
02H	ディスプレイ出力	
03H	補助入力	
04H	補助出力	
05H	プリンター出力.................〇	
06H	直接コンソール入出力	
07H	直接コンソール入力(エコーなし)	
08H	コンソール入力(エコーなし)	
09H	文字列表示	
0AH	バッファー付きキーボード入力	
0BH	標準入力状況のチェック	
0CH	キーボード・バッファーの消去およびキーボード機能の呼び出し	
0DH	ディスク・リセット	
0EH	ディスクの選択	
0FH	ファイルのオープン	
10H	ファイルのクローズ	
11H	最初のエントリーの探索	
12H	次のエントリーの探索	
13H	ファイルの削除	
14H	順次読み取り	
15H	順次書き込み	
16H	ファイルの作成	
17H	ファイル名の変更	
18H	DOSにより予約済み	
19H	現行ディスク	
1AH	ディスク転送アドレス(DTA)の設定	

番号	機能名	互換性の考慮点
1BH	割り振りテーブル情報	
1CH	指定装置の割り振りテーブル情報	
1DH	DOSにより予約済み	
1EH	DOSにより予約済み	
1FH	DOSにより予約済み	
20H	DOSにより予約済み	
21H	ランダム読み取り	
22H	ランダム書き込み	
23H	ファイル・サイズ	
24H	相対レコード・フィールドの設定	
25H	割り込みベクトルの設定	
26H	新しいプログラム・セグメントの作成	
27H	ランダム・ブロック読み取り	
28H	ランダム・ブロック書き込み	
29H	ファイル名解析	
2AH	日付の取得	
2BH	日付の設定	
2CH	時刻の取得	
2DH	時刻の設定	
2EH	ベリファイ・スイッチの設定と解除	
2FH	ディスク転送アドレス(DTA)の取得	
30H	DOSバージョン番号の取得	
31H	常駐のままプロセス終了	
32H	DOSにより予約済み	
33H	システム値の取得および設定	
34H	DOSにより予約済み	
35H	割り込みベクトルの取得	
36H	ディスク空き領域の取得	
37H	DOSにより予約済み	
38H	国別情報の取得および設定	
39H	サブディレクトリーの作成(MKDIR)	
3AH	サブディレクトリーの削除(RMDIR)	
3BH	現行ディレクトリーの変更(CHDIR)	
3CH	ファイルの作成(CREATE)	
3DH	ファイルのオープン	
3EH	ファイル・ハンドルのクローズ	

番号	機能名	互換性の考慮点
3FH	ファイルまたは装置からの読み取り	
40H	ファイルまたは装置への書き込み	
41H	指定ディレクトリーからのファイルの削除(UNLINK)	
42H	ファイル読み書きポインターの移動(LSEEK)	
43H	ファイル・モードの変更(CHMOD)	
44H	装置の入出力制御(IOCtl)	
45H	ファイル・ハンドルの複製(DUP)	
46H	ファイル・ハンドルの強制複製(FORCDUP)	
47H	現行ディレクトリーの取得	
48H	メモリーの割り当て	
49H	割り当てられたメモリーの解放	
4AH	割り当てられたメモリー・ブロックの変更(SETBLOCK)	
4BH	プログラムのロードまたは実行(EXEC)	
4CH	プロセスの終了(EXIT)	
4DH	サブプロセスからのリターン・コードの取得(WAIT)	
4EH	最初に一致するファイルを見つける(FIND FIRST)	
4FH	次に一致するファイルを見つける(FIND NEXT)	
50H	DOSにより予約済み	
51H	DOSにより予約済み	
52H	DOSにより予約済み	
53H	DOSにより予約済み	
54H	ベリファイ(Verify)設定状況の取得	
55H	DOSにより予約済み	
56H	ファイル名の変更	
57H	ファイルの日付および時刻の取得と設定	
58H	DOSにより予約済み	
59H	拡張エラーの取得	
5AH	ユニーク・ファイルの作成	
5BH	新しいファイルの作成	
5CH	ファイル・アクセスのロックまたはロックの解除	

番号	機能名	互換性の考慮点
5DH	DOSにより予約済み	
5E00H	機器名の取得	
5E02H	プリンター・セットアップの設定	
5E03H	プリンター・セットアップの取得	
5F02H	リダイレクション・リスト・エントリーの取得	
5F03H	装置のリダイレクション	
5F04H	リダイレクションの取り消し	
60H	DOSにより予約済み	
61H	DOSにより予約済み	
62H	プログラム・セグメント・プリフィックス(PSP)アドレスの取得	
63H	DBCSベクター情報の取得	
64H	DOSにより予約済み	
65H	拡張国別情報の取得	
66H	グローバル・コード・ページの取得と設定	
67H	ハンドル数の設定	
68H	ファイルの引き渡し	
69H	DOSにより予約済み	
6AH	DOSにより予約済み	
6BH	DOSにより予約済み	
6CH	拡張されたオープンおよび作成	

05H: プリンター出力

アプリケーションは“DOSファンクション・コール 05H: プリンター出力”か“INT 17H”のどちらかを使って出力してください。

文字コードやコントロール・コードなどのデータ・ストリームに関する詳細は、『プリンター・データストリーム』の章を参照してください。

DOS割り込み (INT 20H-3FH)

割り込みタイプとして20Hから3FHまでがDOSに予約されています。

DOS/Vで使用可能なDOS割り込みを以下の表にまとめした。日本語サポートのために、英語版からインターフェースが拡張されたものはありません。

詳細は“DOS/V 技術解説編”を参照して下さい。

割り込みタイプ	名前
INT 20H	プログラムの終了
INT 21H	ファンクション・リクエスト
INT 22H	終了アドレス
INT 23H	Ctrl-Break出口アドレス
INT 24H	重大エラー・ハンドラー・ベクトル
INT 25H/26H	絶対ディスクの読み取り/書き込み
INT 27H	常駐のまま終了
INT 28H-2EH	DOSのために予約済み
INT 2FH	多重割り込み
INT 30H-3FH	DOSのために予約済み

BIOS割り込み (INT 10H-1FH)

アプリケーションから、タイプ10H-1FHの割り込みによってDOSを経由しないで直接BIOSルーチンを呼び出し、低レベルのブロックまたは文字の入出力や装置操作を行うことができます。

DOS/Vで使用可能なBIOS割り込みを以下の表にまとめました。日本語サポートのために英語版から変更を加えたもの、OADGアプリケーションとして考慮すべきものには印を付けました。

本書では、アプリケーション・ポータビリティーを確保するために必要な考慮点について説明します。それ以外の詳細は、“DOS/V BIOS インターフェース技術解説編”と“IBM Personal System/2 and Personal Computer BIOS Interface Technical Reference”の『BIOS Section 2. Interrupts』を参照してください。

割り込みタイプ	名前	2バイトのための拡張	互換性の考慮点	参照資料(*)
INT 10H	ディスプレイ入出力	○		1.
INT 11H	装置構成情報			
INT 12H	記憶域サイズを得る			
INT 13H	ディスケット・ディスク入出力			
INT 14H	ASYNC入出力			
INT 15H	システム・サービス入出力	○	○	1.
INT 16H	キーボード入出力	○	○	
INT 17H	プリンター入出力	○	○	
INT 18H-19H	BIOS用に予約済み			1.
INT 1AH	時刻			1.
INT 1BH	キーボードBREAKアドレス			
INT 1CH	タイマー刻みベクトル			
INT 1DH-1FH	BIOS用に予約済み			

*:前述参考資料の番号

INT 15H: システム・サービス入出力

INT 15Hはハードウェアに近い低位のインターフェースです。上位のインターフェースで同じ機能がある場合、そちらを使ってください。

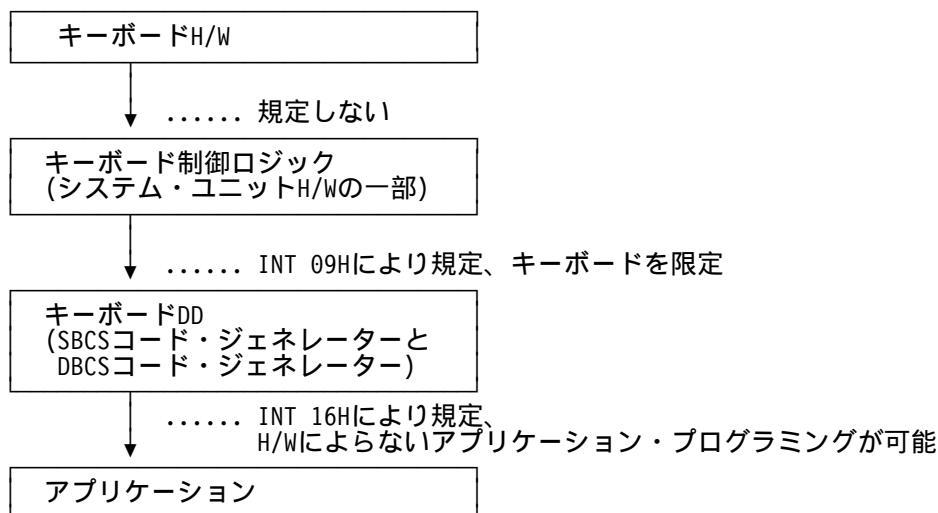
INT 15H AH=4FH: キーボード・インターセプト

この割り込みを取り込む場合、制御を元のアドレスにチェーンしてください。チェーンしないと、キー入力が処理されなくなるものがあります。

INT 15H AH=88H: 拡張メモリー・サイズの判別

RAMフォントを使用する場合、1MB超のメモリー空間にフォントをロードします。このとき、INT 15H AH=88H: 拡張メモリー・サイズの判別で返される1MB超の使用可能なメモリー・サイズを調整することで、フォントのロードされた領域を保護します。DOS/V 5.0では1MB超のメモリーを使用するデバイス・ドライバーなどとの衝突を避けるために、$FONT.SYSはCONFIG.SYSの最初に指定してください。

INT 16H: キーボード入出力

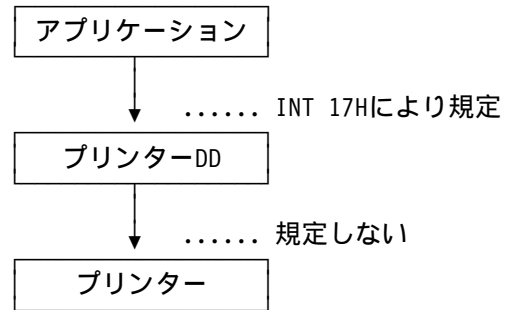
アプリケーションとキーボード・デバイス・ドライバー間のインターフェースを規定し、H/Wインターフェースは規定しません。

アプリケーションは、DOSファンクション・コールまたはBIOSインターフェースINT 16Hで文字コードを読み取ります。文字コードと拡張コードを使用することで異なるキーボードをサポートできます。走査コードは、原則として使いません。

コードに関する詳細は『キーボード入力』および『SBCSコード・ジェネレーター』の章を参照してください。

INT 17H: プリンター入出力

アプリケーションとプリンター・デバイス・ドライバー間のインターフェースを規定し、H/Wインターフェースは規定しません。

アプリケーションはDOSファンクション・コール05H:プリンター出力、またはINT 17Hのどちらかを使って出力してください。

文字コードとコントロール・コードなどのデータ・ストリームに関する詳細は、『プリンター・データストリーム』の章を参照してください。

INT 33H: マウス・ドライバーのマウス機能

マウス機能は“INT 33H”を使います。

詳細は“DOS/Vマウス・ドライバー技術解説編”を参照してください。

BIOSパラメーター・テーブル (INT 1EH)

割り込み1EHはディスク・ドライブのパラメーター・テーブルへの入り口を指し示します。

このテーブルのフォーマット、内容はハードウェアによります。このテーブルを参照するアプリケーションは、ハードウェアの違いを考慮してください。

テーブルの詳細は、 “DOS/V BIOSインターフェース技術解説編”、“IBM Personal System/2 and Personal Computer BIOS Interface Technical Reference”の『BIOS Section 3. Data Areas and ROM Tables』を参照してください。

ビデオ・モード

DOS/VはVGA互換のディスプレイ・アダプターだけで、特別なハードウェアなしに漢字などの2バイト文字を表示することができます。

次のビデオ・モードで2バイト文字を使用できます。

- モード11H　2色カラー・グラフィック・モード　640x480ドット
- モード12H　16色カラー・グラフィック・モード　640x480ドット
- モード72H　16色カラー・グラフィック・モード　640x480ドット

- モード03H　16色エミュレートCGAテキスト・モード　640x475ドット
- モード73H　16色エミュレート拡張CGAテキスト・モード　640x475ドット

モード03Hと73Hは、ソフトウェア・エミュレーションのモードです。実際に使用されるのはカラー・グラフィック・モードですが、アプリケーションに対してはテキスト・モードのインターフェースを提供しています。BIOS割り込みを使って2バイト文字を書き込むことができます。

また、BIOS割り込みINT 10H AH=FEHを使って、テキスト・バッファーのアドレスを得ることができます。このテキスト・バッファーに直接書き込まれた文字は、INT 10H AH=FFHを発行することで更新、表示されます。

INT 10H ディスプレイ入出力を使って文字を出力するとき、文字属性(アトリビュート)を指定できます。また読み取りで、文字属性を読むこともできます。このとき、文字属性は各バイトに対して作用します。2バイト文字を書き込むには、文字属性も2バイト指定しますが、左右に同じ文字属性を指定してください。

この他のVGAビデオ・モードは、2バイト文字を表示することはできません。

ビデオ・モードの仕様、インターフェースは“DOS/V BIOSインターフェース技術解説編”および“DOS/V オプション機能 技術解説編”を参照してください。

キーボード入力

OADGでは、以下のキーボードをサポートします。

- IBM 5576-A01キーボード (106キー)
- IBM U.S.English キーボード (101キー)
- AX*仕様キーボード
- 東芝J-3100 *キーボード

キーボードのハードウェアから上がる走査コードは、キーボードによって異なります。この違いを吸収し、アプリケーション、かな漢字変換プログラムに共通のインターフェースを提供するためにいくつかのドライバーがあります。

アプリケーションは、キーボードBIOS(INT 16H)より文字コードを取得することで、キーボードの種類によらないプログラムを書くことができます。

キーの解釈方法

IBM 5576-A01キーボードはJIS配列、IBM U.S.English, AX, 東芝J-3100キーボードはASCII配列です。” @ [] \などの特殊記号の配置は両者で異なります。(フランスにはAZERTY配列のようにアルファベットの位置が異なるキーボードもあります。)このように複数の異なるキーボードを一つのアプリケーションでサポートするには、走査コードを読まずに、BIOSが走査コードから生成した文字コードを読むことです。これにより、かな漢字変換プログラムに対する特別な処置も不要となります。

次のように順に検査することで、走査コード(AH)/文字コード(AL)を解釈します。以下の記述でXX、YYは00H以外の8ビット・コードを表します。

1. 00/00ならBreakである
2. 00/XXなら文字コード(XXH)である
3. YY/00またはYY/E0なら拡張コード(YY)である
4. YY/XXなら文字コード(XXH)である

文字コードは、制御コード、1バイト文字コード、2バイト文字コードの1バイト目または2バイト目のいずれかです。

1. 前に読んだコードが2バイト文字の1バイト目なら、次のコードは2バイト目
2. 00H-1FH, 7FHなら制御コード
3. DBCSベクターの範囲内(DOS/Vでは、81H-9FH、E0H-FCH)なら、2バイト文字の1バイト目
4. それ以外なら1バイト文字(英数カナ)

1バイトの英数カナ文字は、文字コードと走査コードを取得できますが、走査コードを使うと、それぞれのキーボードに対応する処置が必要になります。

キー配列と走査コード/文字コード

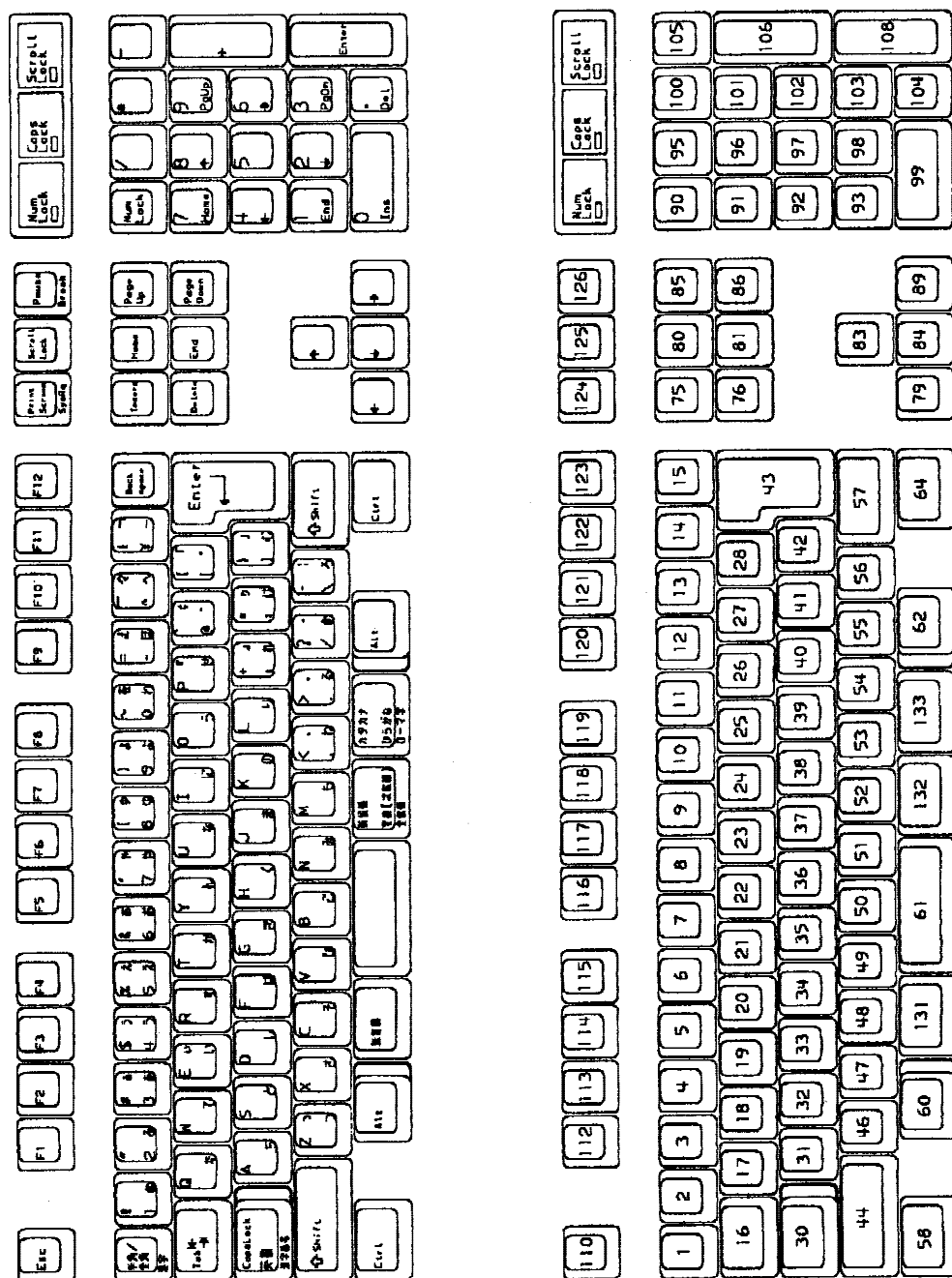



図　1-1. IBM 5576-A01 キーボードとそのキー番号

図　1-2. IBM U.S. Englishキーボードとそのキー番号

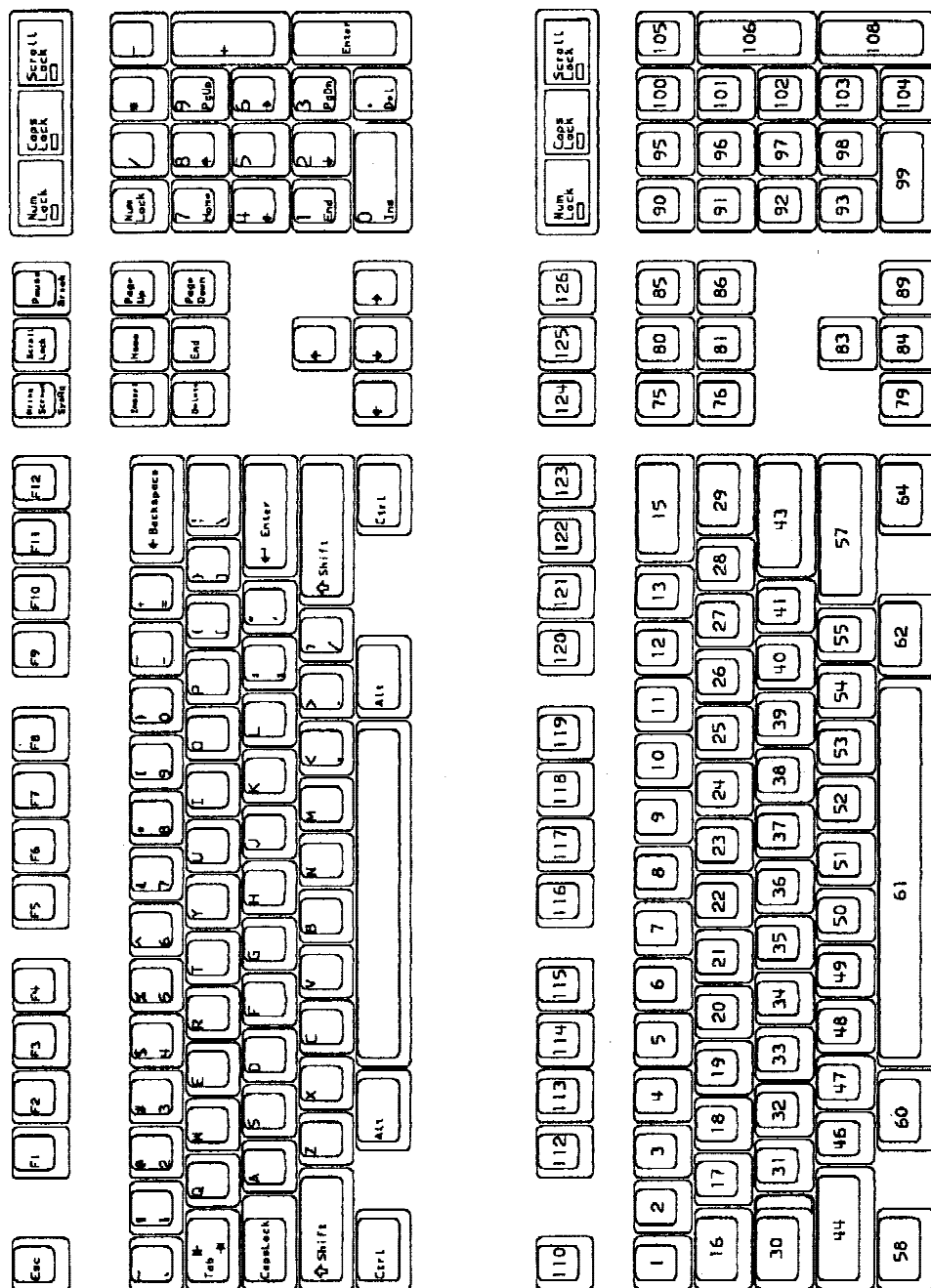
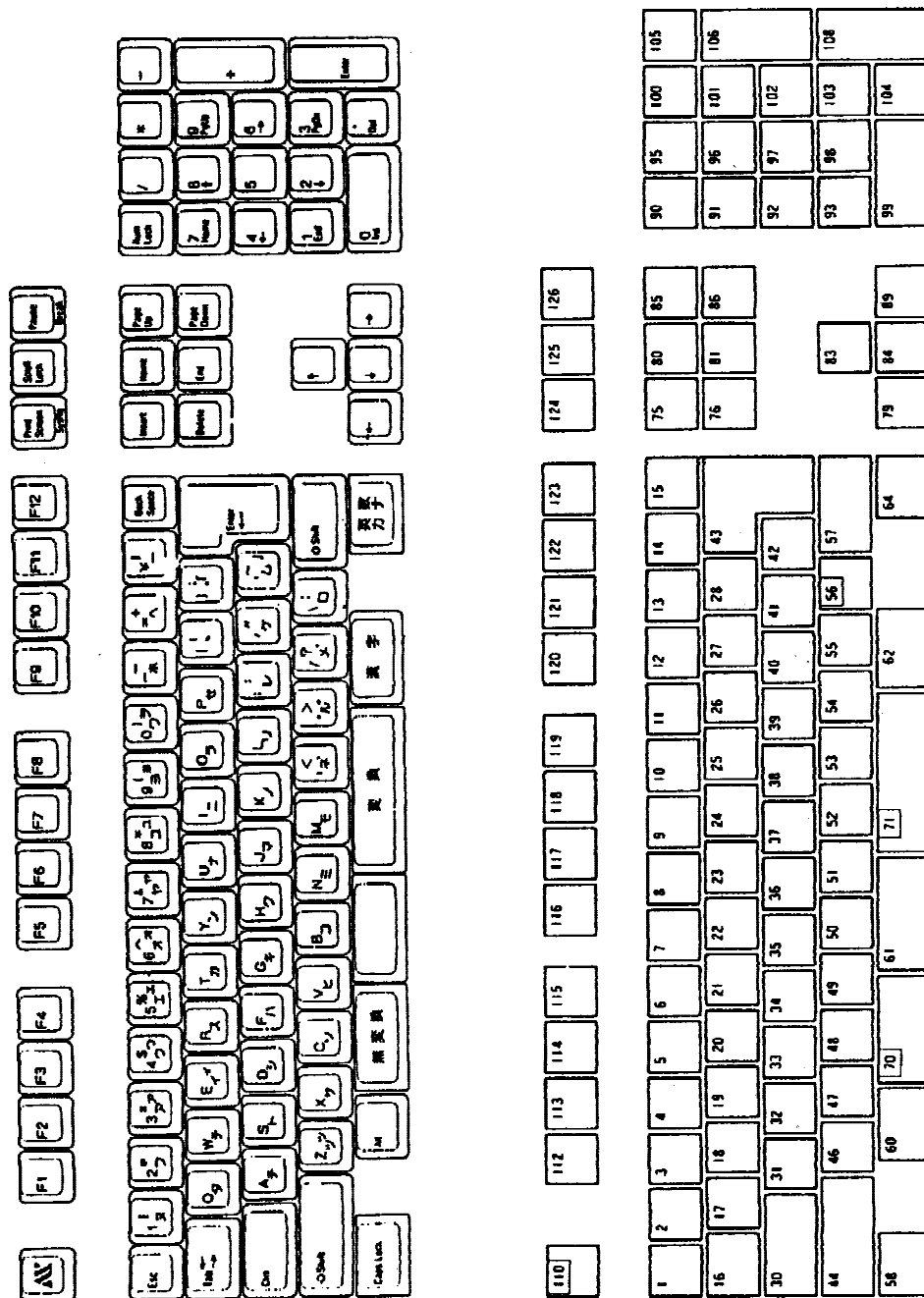



図　1-3. AXキーボードとそのキー番号

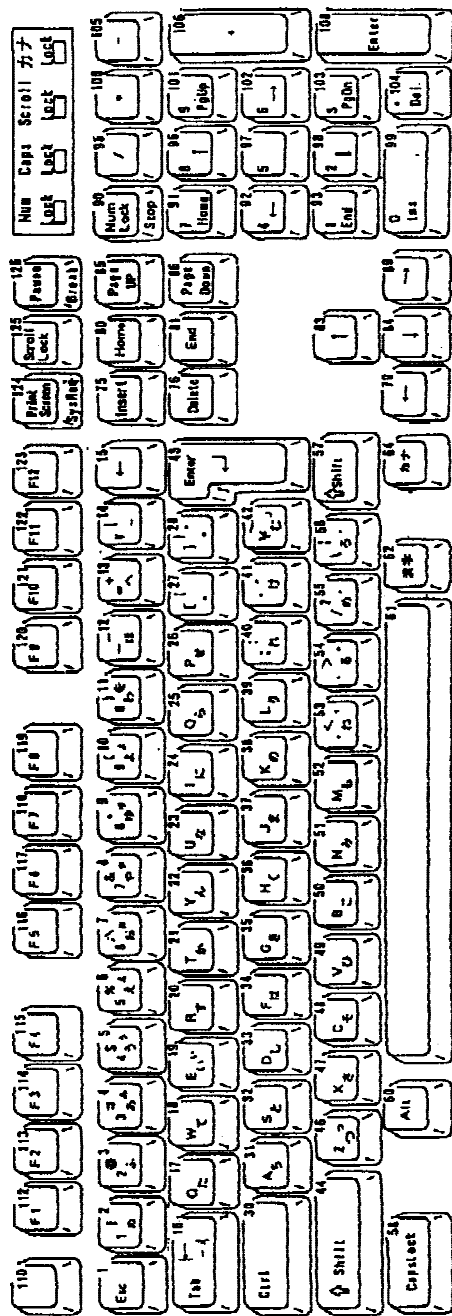



図　1-4. 東芝J-3100キーボードとそのキー番号

文字コードでとれる文字と対応するキーを以下に掲げます。注に関しては、『マニュアル記述上の考慮』で説明しています。

文字	走査/文字コード	キー番号	刻印	キーの表し方	注
SOH	xx/01	C+ 31	A	Ctrl+A	
STX	xx/02	C+ 50	B	Ctrl+B	
ETX	xx/03	C+ 48	C	Ctrl+C	
EOT	xx/04	C+ 33	D	Ctrl+D	
ENQ	xx/05	C+ 19	E	Ctrl+E	
ACK	xx/06	C+ 34	F	Ctrl+F	
BEL	xx/07	C+ 35	G	Ctrl+G	
BS	xx/08	15	Backspace	Backspace	
BS	xx/08	S+ 15	Backspace	Shift+Backspace	
BS	xx/08	C+ 36	H	Ctrl+H	
HT	xx/09	16	Tab Backtab	Tab	
HT	xx/09	C+ 24	I	Ctrl+I	
LF	xx/0A	C+ 43	Enter	Ctrl+Enter	
LF	xx/0A	C+ 37	J	Ctrl+J	
LF	xx/0A, E0/0A	C+108	Enter	Ctrl+Enter	
VT	xx/0B	C+ 38	K	Ctrl+K	
FF	xx/0C	C+ 39	L	Ctrl+L	
CR	xx/0D	43	Enter	Enter	
CR	xx/0D	S+ 43	Enter	Shift+Enter	
CR	xx/0D	C+ 52	M	Ctrl+M	
CR	xx/0D, E0/0D	108	Enter	Enter	
CR	xx/0D, E0/0D	S+108	Enter	Shift+Enter	
SO	xx/0E	C+ 51	N	Ctrl+N	
SI	xx/0F	C+ 25	O	Ctrl+O	
DLE	xx/10	C+ 26	P	Ctrl+P	
DC1	xx/11	C+ 17	Q	Ctrl+Q	
DC2	xx/12	C+ 20	R	Ctrl+R	
DC3	xx/13	C+ 32	S	Ctrl+S	
DC4	xx/14	C+ 21	T	Ctrl+T	
NAK	xx/15	C+ 23	U	Ctrl+U	
SYN	xx/16	C+ 49	V	Ctrl+V	
ETB	xx/17	C+ 18	W	Ctrl+W	
CAN	xx/18	C+ 47	X	Ctrl+X	
EM	xx/19	C+ 22	Y	Ctrl+Y	
SUB	xx/1A	C+ 46	Z	Ctrl+Z	
ESC	xx/1B	110	Esc	Esc	1
ESC	xx/1B	S+110	Esc	Shift+Esc	1
ESC	xx/1B	C+110	Esc	Ctrl+Esc	1
ESC	xx/1B	C+ 28	[	Ctrl+ [	1
FS	xx/1C	C+	＼(または¥)	Ctrl+＼(または¥)	2
GS	xx/1D	C+ 42	]	Ctrl+]	1
RS	xx/1E	C+ 7	6	Ctrl+6	
US	xx/1F	C+ 12	—	Ctrl+—	
SP	xx/20	61	Space	Space	
SP	xx/20	S+ 61	Space	Shift+Space	
SP	xx/20	C+ 61	Space	Ctrl+Space	
SP	xx/20	A+ 61	Space	Alt+Space	
!	xx/21		!	!	
"	xx/22		"	"	1
#	xx/23		#	#	
$	xx/24		$	$	
%	xx/25		%	%	
&	xx/26		&	&	
'	xx/27		'	'	1
(	xx/28		(	(	1

文字	走査/文字コード	キー番号	刻印	キーの表し方	注
)	xx/29		)	)	1
*	xx/2A		*	*	1
*	xx/2A	100	*	*(テンキー)	
*	xx/2A	S+100	*	Shift+*(テンキー)	
+	xx/2B		+	+	1
+	xx/2B	106	+	+(テンキー)	
+	xx/2B	S+106	+	Shift++(テンキー)	
,	xx/2C		,	,	
—	xx/2D		—	—	
—	xx/2D	105	—	—(テンキー)	
—	xx/2D	S+105	—	Shift+—(テンキー)	
.	xx/2E		.	.	
.	xx/2E	S+104	Del .	Shift+Del(テンキー)	
／	xx/2F		／	／	
／	xx/2F, E0/2F	95	／	／(テンキー)	
／	xx/2F, E0/2F	S+ 95	／	Shift+／(テンキー)	
0	xx/30		0	0	
0	xx/30	S+ 99	Ins 0	Shift+Ins(テンキー)	
1	xx/31		1	1	
1	xx/31	S+ 93	End 1	Shift+End(テンキー)	
2	xx/32		2	2	
2	xx/32	S+ 98	↓ 2	Shift+↓(テンキー)	
3	xx/33		3	3	
3	xx/33	S+103	PgDn 3	Shift+PgDn(テンキー)	
4	xx/34		4	4	
4	xx/34	S+ 92	← 4	Shift+←(テンキー)	
5	xx/35		5	5	
5	xx/35	S+ 97	5	Shift+5(テンキー)	
6	xx/36		6	6	
6	xx/36	S+102	→ 6	Shift+→(テンキー)	
7	xx/37		7	7	
7	xx/37	S+ 91	Home 7	Shift+Home(テンキー)	
8	xx/38		8	8	
8	xx/38	S+ 96	↑ 8	Shift+↑(テンキー)	
9	xx/39		9	9	
9	xx/39	S+101	PgUp 9	Shift+PgUp(テンキー)	
:	xx/3A		:	:	1
;	xx/3B		;	;	
<	xx/3C		<	<	
=	xx/3D		=	=	1
>	xx/3E		>	>	
?	xx/3F		?	?	
@	xx/40		@	@	1
A	xx/41	S+	A	A	
B	xx/42	S+	B	B	
C	xx/43	S+	C	C	
D	xx/44	S+	D	D	
E	xx/45	S+	E	E	
F	xx/46	S+	F	F	
G	xx/47	S+	G	G	
H	xx/48	S+	H	H	
I	xx/49	S+	I	I	
J	xx/4A	S+	J	J	
K	xx/4B	S+	K	K	
L	xx/4C	S+	L	L	
M	xx/4D	S+	M	M	
N	xx/4E	S+	N	N	
O	xx/4F	S+	O	O	

文字	走査/文字コード	キー番号	刻印	キーの表し方	注
P	xx/50	S+	P	P	
Q	xx/51	S+	Q	Q	
R	xx/52	S+	R	R	
S	xx/53	S+	S	S	
T	xx/54	S+	T	T	
U	xx/55	S+	U	U	
V	xx/56	S+	V	V	
W	xx/57	S+	W	W	
X	xx/58	S+	X	X	
Y	xx/59	S+	Y	Y	
Z	xx/5A	S+	Z	Z	
[	xx/5B		[	[	1
¥	xx/5C		¥または＼	¥または＼	2
]	xx/5D		]	]	1
^	xx/5E		^	^	1
_	xx/5F		_	_	1
`	xx/60		`	`	1
a	xx/61		A	a	
b	xx/62		B	b	
c	xx/63		C	c	
d	xx/64		D	d	
e	xx/65		E	e	
f	xx/66		F	f	
g	xx/67		G	g	
h	xx/68		H	h	
i	xx/69		I	i	
j	xx/6A		J	j	
k	xx/6B		K	k	
l	xx/6C		L	l	
m	xx/6D		M	m	
n	xx/6E		N	n	
o	xx/6F		O	o	
p	xx/70		P	p	
q	xx/71		Q	q	
r	xx/72		R	r	
s	xx/73		S	s	
t	xx/74		T	t	
u	xx/75		U	u	
v	xx/76		V	v	
w	xx/77		W	w	
x	xx/78		X	x	
y	xx/79		Y	y	
z	xx/7A		Z	z	
{	xx/7B		[{	{	1
|	xx/7C		| または¦	| または¦	2
}	xx/7D		] }	}	1
‾	xx/7E		‾ または～	‾ または～	2
DEL	xx/7F	C+ 15	Backspace	Ctrl+Backspace	

その他の文字コード		意味	
xx/80		未定義の1バイト文字	
00/81-00/9F		2バイト文字の1バイト目	
xx/A0		未定義の1バイト文字	
xx/A1-xx/DF		1バイト・カタカナ	
00/E0-00/FC		2バイト文字の1バイト目	
xx/FD-xx/FF		未定義の1バイト文字	

拡張コードでとれる機能と対応するキーを以下に掲げます。注に関しては、『マニュアル記述上の考慮』で説明しています。

拡張コード	共通または 5576-A01 キー番号　刻印	IBM U.S. English (101), AX, J-3100の違い	注
00/00	C+126 Pause (Break)		
/, 01/00	A+ Esc		1
NUL 03/00	C+ 3 2		
/, 0E/00	A+ 15 Backspace		
0F/00	S+ 16 Tab Backtab		
10/00	A+ 17 Q		
11/00	A+ 18 W		
12/00	A+ 19 E		
13/00	A+ 20 R		
14/00	A+ 21 T		
15/00	A+ 22 Y		
16/00	A+ 23 U		
17/00	A+ 24 I		
18/00	A+ 25 O		
19/00	A+ 26 P		
/, 1A/00	A+ 27 @	A+ 27 [(101, AX, J-3100)	3
/, 1B/00	A+ 28 [	A+ 28] (101, AX, J-3100)	3
/, 1C/00	A+ 43 Enter		
1E/00	A+ 31 A		
1F/00	A+ 32 S		
20/00	A+ 33 D		
21/00	A+ 34 F		
22/00	A+ 35 G		
23/00	A+ 36 H		
24/00	A+ 37 J		
25/00	A+ 38 K		
26/00	A+ 39 L		
/, 27/00	A+ 40 ;		
/, 28/00	A+ 41 :	A+41 ` (101, AX, J-3100)	3
/, 2B/00	A+ 42]	A+29 ＼ (101), A+14 ¥ (AX, J-3100)	4
2C/00	A+ 46 Z		
2D/00	A+ 47 X		
2E/00	A+ 48 C		
2F/00	A+ 49 V		
30/00	A+ 50 B		
31/00	A+ 51 N		
32/00	A+ 52 M		
/, 33/00	A+ 53 ,		
/, 34/00	A+ 54 .		
/, 35/00	A+ 55 ／		
/, 37/00	A+100 *		
3B/00	112 F1		
3C/00	113 F2		
3D/00	114 F3		
3E/00	115 F4		
3F/00	116 F5		
40/00	117 F6		
41/00	118 F7		
42/00	119 F8		
43/00	120 F9		
44/00	121 F10		
47/00	91 Home 7		
47/00, 47/E0	80 Home		
47/00, 47/E0	S+ 80 Home		

拡張コード	共通または 5576-A01 キー番号　刻印	IBM U.S. English (101), AX, J-3100の違い	注
48/00	96 ↑ 8		
48/00, 48/E0	83 ↑		
48/00, 48/E0	S+ 83 ↑		
49/00	101 PgUp 9		
49/00, 49/E0	85 PageUp		
49/00, 49/E0	S+ 85 PageUp		
/, 4A/00	A+105 —		
4B/00	92 ← 4		
4B/00, 4B/E0	79 ←		
4B/00, 4B/E0	S+ 79 ←		
/, 4C/00	97 5		
4D/00	102 → 6		
4D/00, 4D/E0	89 →		
4D/00, 4D/E0	S+ 89 →		
/, 4E/00	A+106 +		
4F/00	93 End 1		
4F/00, 4F/E0	81 End		
4F/00, 4F/E0	S+ 81 End		
50/00	98 ↓ 2		
50/00, 50/E0	84 ↓		
50/00, 50/E0	S+ 84 ↓		
51/00	103 PgDn 3		
51/00, 51/E0	86 PageDown		
51/00, 51/E0	S+ 86 PageDown		
52/00	99 Ins 0		
52/00, 52/E0	75 Insert		
52/00, 52/E0	S+ 75 Insert		
53/00	104 Del .		
53/00, 53/E0	76 Delete		
53/00, 53/E0	S+ 76 Delete		
54/00	S+112 F1		
55/00	S+113 F2		
56/00	S+114 F3		
57/00	S+115 F4		
58/00	S+116 F5		
59/00	S+117 F6		
5A/00	S+118 F7		
5B/00	S+119 F8		
5C/00	S+120 F9		
5D/00	S+121 F10		
5E/00	C+112 F1		
5F/00	C+113 F2		
60/00	C+114 F3		
61/00	C+115 F4		
62/00	C+116 F5		
63/00	C+117 F6		
64/00	C+118 F7		
65/00	C+119 F8		
66/00	C+120 F9		
67/00	C+121 F10		
68/00	A+112 F1		
69/00	A+113 F2		
6A/00	A+114 F3		
6B/00	A+115 F4		
6C/00	A+116 F5		
6D/00	A+117 F6		
6E/00	A+118 F7		
6F/00	A+119 F8		
70/00	A+120 F9		
71/00	A+121 F10		
72/00	C+124 PrintScreen		

拡張コード	共通または 5576-A01 キー番号 刻印	IBM U.S. English (101), AX, J-3100の違い	注
73/00	C+ 92 ← 4		
73/00, 73/E0	C+ 79 ←		
74/00	C+102 → 6		
74/00, 74/E0	C+ 89 →		
75/00	C+ 93 End 1		
75/00, 75/E0	C+ 81 End		
76/00	C+103 PgDn 3		
76/00, 76/E0	C+ 86 PageDown		
77/00	C+ 91 Home 7		
77/00, 77/E0	C+ 80 Home		
78/00	A+ 2 1		
79/00	A+ 3 2		
7A/00	A+ 4 3		
7B/00	A+ 5 4		
7C/00	A+ 6 5		
7D/00	A+ 7 6		
7E/00	A+ 8 7		
7F/00	A+ 9 8		
80/00	A+ 10 9		
81/00	A+ 11 0		
82/00	A+ 12 —		
83/00	A+ 13 ^	A+ 13 = (101, AX, J-3100)	3
84/00	C+101 PgUp 9		
84/00, 84/E0	C+ 85 PageUp		
/, 85/00	122 F11		
/, 86/00	123 F12		
/, 87/00	S+122 F11		
/, 88/00	S+123 F12		
/, 89/00	C+122 F11		
/, 8A/00	C+123 F12		
/, 8B/00	A+122 F11		
/, 8C/00	A+123 F12		
/, 8D/00	C+ 96 ↑ 8		
/, 8D/E0	C+ 83 ↑		
/, 8E/00	C+105 —		
/, 8F/00	C+ 97 5		
/, 90/00	C+106 +		
/, 91/00	C+ 98 ↓ 2		
/, 91/E0	C+ 84 ↓		
/, 92/00	C+ 99 Ins 0		
/, 92/E0	C+ 75 Insert		
/, 93/00	C+104 Del .		
/, 93/E0	C+ 76 Delete		
/, 94/00	C+ 16 Tab Backtab		
/, 95/00	C+ 95 /		
/, 96/00	C+100 *		
/, 97/00	A+ 80 Home		
/, 98/00	A+ 83 ↑		
/, 99/00	A+ 85 PageUp		
/, 9B/00	A+ 79 ←		
/, 9D/00	A+ 89 →		
/, 9F/00	A+ 81 End		
/, A0/00	A+ 84 ↓		
/, A1/00	A+ 86 PageDown		
/, A2/00	A+ 75 Insert		
/, A3/00	A+ 76 Delete		
/, A4/00	A+ 95 /		
/, A5/00	A+ 16 Tab Backtab		
/, A6/00	A+108 Enter		

拡張コード

通常のJIS8ビット・コードでは表せない機能のために拡張コードが用いられます。ALに00HまたはE0Hが返され、AHに疑似走査コードが返されます。拡張コードを使う場合の考慮点を掲げます。

- 標準のキー読み取り(INT 16H AH=00H/01H)では拡張コードが返らないキーも多数あります。Ctrlキー、Altキーを多用するアプリケーションは拡張機能(INT 16H AH=10H/11H)を用いて拡張コードを読み取ってください。
- カーソル・キーとテンキーの下段のカーソルは、標準のキー読み取りでは同じ拡張コードですが、拡張機能ではXX/00とXX/E0として区別できます。
- Alt+インボード・キー(キー番号2-14, 17-29, 31-42, 45-56)は通常“Alt+A”などと記述しますが、拡張コードが疑似走査コードのため、複数のキーボードを考えると、記述が異なることがあります。たとえば、Pの右のキー(#27)は5576-A01では@、U.S.キーボードでは[です。それで、同じ拡張コードが上がる位置のキーをそれぞれ、Alt+@、Alt+[と別の呼び方をすることになります。表の注3のキーが、4つのキーボード間で呼び方が変わるキーです。この問題を避けるには、このキーを使わないか、キーボードごとの説明を付けるか、キーボードに因らない命名(Alt+#27など)をしてください。将来フランスなどのキーボードをサポートする場合にはAからZのキーについても考慮が必要です。
- アウトボードのファンクション・キー、カーソル・キー、テン・キー、及びこれらのキーとShift、Ctrl、Altのシフト状態の組み合わせのキーは共通です。
- あるキーボードに固有のキーを使う必要がある場合、他のキーボードに対しては同じ機能を割り当てる代替キーを確保してください。

マニュアル記述上の考慮

配列の異なる4つのキーボードをマニュアルに記述するとき、以下の点を考慮してすべてのユーザーに分かりやすいキー説明をしてください。

- 文字キーを示すときはシフトを付けません。たとえば、@を表すには単に“@”キーとしてください。これは特殊記号が上段にあるキーボードと、下段にあるものがあるからです。U.S.キーボード用に“Shift+2”と書くと、5576-A01キーボードでは“"”になってしまいます。
- インボードの特殊記号、Esc、などはキーボードによって位置が異なります(表の注1)。しかし、キートップの刻印に対応して同じ文字コードや拡張コードが上がります。このようなキーは、刻印で表してください。
- コードページの違いのため異なる刻印のキーで入力することがあります。(表の注2)。このような場合、両方を併記して表します。たとえば、制御コード1C(FS)を入力するにはCtrl+\またはCtrl+¥とします。
- Altシフトによる拡張コードは疑似走査コードなので、キートップの刻印の文字とは無関係にキー番号で決ります。(表の注3)このようなキーに対しては、
  - “Alt+キー27”のようにキー番号に対応した命名をする

- キーボードごとのキーを表にまとめ、機能名で参照する
- 使わない
- キーボードごとのキーを列挙する
などの考慮をしてください。

将来、ヨーロッパ圏のキーボード(IBM 102 キーボードなど)をサポートする場合は、アルファベット・キーも考慮が必要です。

- かな漢字変換を日本語入力サブシステムにまかせているアプリケーションは、かな漢字変換に関するキーについては記述しないほうがよいでしょう。FEPによって使用するキーやその機能が異なるからです。

- 表の注4の拡張コード(2B/00)は、各キーボードでキー位置も刻印も異なります。できれば他のキーで代用してください。

SBCSコード・ジェネレーター

SBCSコード・ジェネレーターの入力は、INT 09Hです。

各キーに対する走査コードは、キーボードによって異なります。OADGでは、4つのキーボードとその走査コードを定義しています。コード表は、“OADGハードウェア・インターフェース技術解説編”を参照してください。

SBCSコード・ジェネレーターは次のシフト状態を管理します。

- Shift
- Ctrl
- Alt
- Caps Lock
- Scroll Lock
- Num Lock

現在のシフト状態と、押されたキーから文字コードを生成します。たとえば、

- Shiftが押されていて、“a”のキーが押されたら、“A”を表す文字コード(41H)を生成します。
- CapsLockモードで、“a”のキーが押されたら、“A”を表す文字コード(41H)を生成します。
- Ctrlキーが押されていて、“a”のキーが押されたら、“Ctrl-A(SOH)”を表す文字コード(01H)を生成します。
- Altキーが押されていて、“a”のキーが押されたら、“Alt-A”を表す拡張コード(1EH/00H)を生成します。

SBCSコード・ジェネレーターの出力は、走査コードと文字コードの組み合わせ、または拡張コードです。BIOSキー・バッファーに保管され、INT 16Hで読み込まれます。

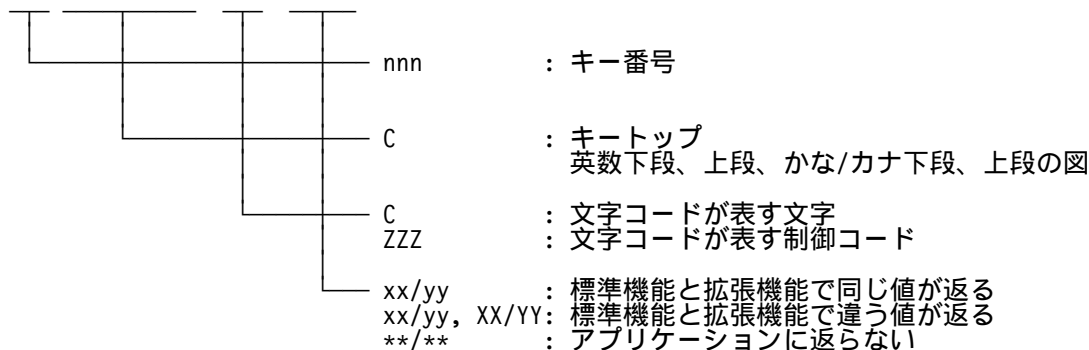
キー・コード表

SBCSコード・ジェネレーターの出力を、表にまとめます。表において2桁の16進数は1バイト・コードを表します。通常、走査コードと文字コードの2つのコードが返されます。“標準機能(INT 16H, AH=00H/01H)”と“拡張機能(INT 16H, AH=10H/11H)”により返される組み合わせが、同じものと、異なるものがあります。表中の走査コード/文字コードの組み合わせが1つのものは、両機能で同じコードが返されます。2つのものは、左側が標準機能、右側が拡張機能で返されるコードです。

DBCSコード・ジェネレーターが導入された後では、通常次のような日本語入力関係のキーのコードは返りません。

> 漢字、英数、カタカナ、ひらがな、カナ、英数カナ、変換、無変換、半角/全角、ローマ字、漢字番号、全候補、前候補、漢字制御、IBM U.S. EnglishキーボードのAlt+ ` (キー番号1)

注:

1. xx/yyは走査コード/文字コードの組み合わせを16進数で示します。
2. xx/00またはxx/E0は、拡張コードを示します。
3. xx/yy, XX/YYはそれぞれ標準機能、拡張機能で返されるコードです。
4. **/**はアプリケーションに渡らないキーを示します。

IBM 5576-A01 キーボード

キー番号	刻印	下段	上段 (Shift+)	Ctrl+	Alt+
1	半角/全角	**/**, AF/00	**/**, B0/00	**/**, B1/00	**/**, B2/00
2	1 ! ぬ	1 02/31	! 02/21	**/**	78/00
3	2 " ふ	2 03/32	" 03/22	NUL 03/00	79/00
4	3 # あ ぁ	3 04/33	# 04/23	**/**	7A/00
5	4 $ う ぅ	4 05/34	$ 05/24	**/**	7B/00
6	5 % え ぇ	5 06/35	% 06/25	**/**	7C/00
7	6 & お ぉ	6 07/36	& 07/26	RS 07/1E	7D/00
8	7 ' や ゃ	7 08/37	' 08/27	**/**	7E/00
9	8 (ゆ ゅ	8 09/38	(09/28	**/**	7F/00
10	9) よ ょ	9 0A/39	) 0A/29	**/**	80/00
11	0 ～ わ を	0 0B/30	**/**, 0B/00	**/**	81/00
12	－ ＝ ほ £	－ 0C/2D	＝ 0C/3D	US 0C/1F	82/00
13	^ ￣ へ 々	^ 0D/5E	￣ 00/7E	**/**	83/00
14	¥ | ー ¬	¥ 7D/5C	| 7D/7C	FS 7D/1C	**/**
15	Backspace	BS 0E/08	BS 0E/08	DEL 0E/7F	**/**, 0E/00
16	Tab Backtab	HT 0F/09	0F/00	**/**, 94/00	**/**, A5/00
17	Q た	q 10/71	Q 10/51	DC1 10/11	10/00
18	W て	w 11/77	W 11/57	ETB 11/17	11/00
19	E い ぃ	e 12/65	E 12/45	ENQ 12/05	12/00
20	R す	r 13/72	R 13/52	DC2 13/12	13/00
21	T か	t 14/74	T 14/54	DC4 14/14	14/00
22	Y ん	y 15/79	Y 15/59	EM 15/19	15/00
23	U な	u 16/75	U 16/55	NAK 16/15	16/00
24	I に	i 17/69	I 17/49	HT 17/09	17/00
25	O ら	o 18/6F	O 18/4F	SI 18/0F	18/00
26	P せ 「	p 19/70	P 19/50	DLE 19/10	19/00
27	@ ` ゛ ¢	@ 1A/40	` 1A/60	**/**	**/**, 1A/00
28	[{ ゜ 「	[1B/5B	{ 1B/7B	ESC 1B/1B	**/**, 1B/00
29	(キーなし)				
30	英数 CapsLock	**/**, B3/00	**/**	**/**, B4/00	**/**, B5/00
31	A ち	a 1E/61	A 1E/41	SOH 1E/01	1E/00
32	S と	s 1F/73	S 1F/53	DC3 1F/13	1F/00
33	D し	d 20/64	D 20/44	EOT 20/04	20/00
34	F は	f 21/66	F 21/46	ACK 21/06	21/00
35	G き	g 22/67	G 22/47	BEL 22/07	22/00
36	H く	h 23/68	H 23/48	BS 23/08	23/00
37	J ま	j 24/6A	J 24/4A	LF 24/0A	24/00
38	K の	k 25/6B	K 25/4B	VT 25/0B	25/00
39	L り	l 26/6C	L 26/4C	FF 26/0C	26/00
40	; ＋ れ 』	; 27/3B	＋ 27/2B	**/**	**/**, 27/00
41	: ＊ け ヶ	: 28/3A	＊ 28/2A	**/**	**/**, 28/00
42	] } む 」	] 2B/5D	} 2B/7D	GS 2B/1D	**/**, 2B/00
43	Enter	CR 1C/0D	CR 1C/0D	LF 1C/0A	**/**, 1C/00
44	Shift	**/**	**/**	**/**	**/**
45	(キーなし)				
46	Z つ っ	z 2C/7A	Z 2C/5A	SUB 2C/1A	2C/00
47	X さ	x 2D/78	X 2D/58	CAN 2D/18	2D/00
48	C そ	c 2E/63	C 2E/43	ETX 2E/03	2E/00
49	V ひ	v 2F/76	V 2F/56	SYN 2F/16	2F/00
50	B こ	b 30/62	B 30/42	STX 30/02	30/00
51	N み	n 31/6E	N 31/4E	SO 31/0E	31/00
52	M も	m 32/6D	M 32/4D	CR 32/0D	32/00
53	, ＜ ね 、	, 33/2C	＜ 33/3C	**/**	**/**, 33/00
54	. ＞ る 。	. 34/2E	＞ 34/3E	**/**	**/**, 34/00
55	／ ? め ・	／ 35/2F	? 35/3F	**/**	**/**, 35/00
56	＼ _ ろ ¦	¥ 73/5C	_ 73/5F	FS 73/1C	**/**
57	Shift	**/**	**/**	**/**	**/**
58	Ctrl	**/**	**/**	**/**	**/**
59	(キーなし)				
60	Alt	**/**	**/**	**/**	**/**
61	Space	SP 39/20	SP 39/20	SP 39/20	SP 39/20
62	Alt	**/**	**/**	**/**	**/**
63	(キーなし)				
64	Ctrl	**/**	**/**	**/**	**/**
65	(キーなし)				
66	(キーなし)				
67	(キーなし)				

IBM 5576-A01 キーボード (続き)

キー番号 刻印	下段	上段 (Shift+)	Ctrl+	Alt+
68 (キーなし)				
69 (キーなし)				
70 (キーなし)				
71 (キーなし)				
72 (キーなし)				
73 (キーなし)				
74 (キーなし)				
75 Insert	52/00, 52/E0	52/00, 52/E0	**/**, 92/E0	**/**, A2/00
76 Delete	53/00, 53/E0	53/00, 53/E0	**/**, 93/E0	**/**, A3/00
77 (キーなし)				
78 (キーなし)				
79 ←	4B/00, 4B/E0	4B/00, 4B/E0	73/00, 73/E0	**/**, 9B/00
80 Home	47/00, 47/E0	47/00, 47/E0	77/00, 77/E0	**/**, 97/00
81 End	4F/00, 4F/E0	4F/00, 4F/E0	75/00, 75/E0	**/**, 9F/00
82 (キーなし)				
83 ↑	48/00, 48/E0	48/00, 48/E0	**/**, 8D/E0	**/**, 98/00
84 ↓	50/00, 50/E0	50/00, 50/E0	**/**, 91/E0	**/**, A0/00
85 PageUp	49/00, 49/E0	49/00, 49/E0	84/00, 84/E0	**/**, 99/00
86 PageDown	51/00, 51/E0	51/00, 51/E0	76/00, 76/E0	**/**, A1/00
87 (キーなし)				
88 (キーなし)				
89 →	4D/00, 4D/E0	4D/00, 4D/E0	74/00, 74/E0	**/**, 9D/00
90 NumLock	**/**	**/**	**/**	**/**
91 Home 7	47/00	7 47/37	77/00	Code Input
92 ← 4	4B/00	4 4B/34	73/00	Code Input
93 End 1	4F/00	1 4F/31	75/00	Code Input
94 (キーなし)				
95 ／	／ 35/2F, E0/2F	／ 35/2F, E0/2F	**/**, 95/00	**/**, A4/00
96 ↑ 8	48/00	8 48/38	**/**, 8D/00	Code Input
97 5	**/**, 4C/00	5 4C/35	**/**, 8F/00	Code Input
98 ↓ 2	50/00	2 50/32	**/**, 91/00	Code Input
99 Ins 0	52/00	0 52/30	**/**, 92/00	Code Input
100 *	* 37/2A	* 37/2A	**/**, 96/00	**/**, 37/00
101 PgUp 9	49/00	9 49/39	84/00	Code Input
102 → 6	4D/00	6 4D/36	74/00	Code Input
103 PgDn 3	51/00	3 51/33	76/00	Code Input
104 Del .	53/00	. 53/2E	**/**, 93/00	**/**
105 −	− 4A/2D	− 4A/2D	**/**, 8E/00	**/**, 4A/00
106 +	+ 4E/2B	+ 4E/2B	**/**, 90/00	**/**, 4E/00
107 (キーなし)				
108 Enter	CR 1C/0D, E0/0D	CR 1C/0D, E0/0D	LF 1C/0A, E0/0A	**/**, A6/00
109 (キーなし)				
110 Esc	ESC 01/1B	ESC 01/1B	ESC 01/1B	**/**, 01/00
111 (キーなし)				
112 F1	3B/00	54/00	5E/00	68/00
113 F2	3C/00	55/00	5F/00	69/00
114 F3	3D/00	56/00	60/00	6A/00
115 F4	3E/00	57/00	61/00	6B/00
116 F5	3F/00	58/00	62/00	6C/00
117 F6	40/00	59/00	63/00	6D/00
118 F7	41/00	5A/00	64/00	6E/00
119 F8	42/00	5B/00	65/00	6F/00
120 F9	43/00	5C/00	66/00	70/00
121 F10	44/00	5D/00	67/00	71/00
122 F11	**/**, 85/00	**/**, 87/00	**/**, 89/00	**/**, 8B/00
123 F12	**/**, 86/00	**/**, 88/00	**/**, 8A/00	**/**, 8C/00
124 PrintScreen	**/**	**/**	72/00	**/**
125 ScrollLock	**/**	**/**	**/**	**/**
126 Pause	**/**	**/**	00/00	**/**
127 (キーなし)				
128 (キーなし)				
129 (キーなし)				
130 (キーなし)				
131 無変換	**/**, AB/00	**/**, AC/00	**/**, AD/00	**/**, AE/00
132 変換 前候補	**/**, A7/00	**/**, A8/00	**/**, A9/00	**/**, AA/00
133 ひらがな カタカナ	**/**, B6/00	**/**, B7/00	**/**, B8/00	**/**, B9/00

IBM U.S. English キーボード

キー番号	刻印	下段	上段 (Shift+)	Ctrl+	Alt+
1	` ~	` 29/60	29/7E	**/**	**/**, 29/00
2	1 !	1 02/31	! 02/21	**/**	78/00
3	2 @	2 03/32	@ 03/40	NUL 03/00	79/00
4	3 #	3 04/33	# 04/23	**/**	7A/00
5	4 $	4 05/34	$ 05/24	**/**	7B/00
6	5 %	5 06/35	% 06/25	**/**	7C/00
7	6 ^	6 07/36	^ 07/5E	RS 07/1E	7D/00
8	7 &	7 08/37	& 08/26	**/**	7E/00
9	8 *	8 09/38	* 09/2A	**/**	7F/00
10	9 (	9 0A/39	(0A/28	**/**	80/00
11	0)	0 0B/30	) 0B/29	**/**	81/00
12	− ＿	− 0C/2D	＿ 0C/5F	US 0C/1F	82/00
13	= +	= 0D/3D	+ 0D/2B	**/**	83/00
14	(キーなし)				
15	Backspace	BS 0E/08	BS 0E/08	DEL 0E/7F	**/**, 0E/00
16	Tab Backtab	HT 0F/09	0F/00	**/**, 94/00	**/**, A5/00
17	Q	q 10/71	Q 10/51	DC1 10/11	10/00
18	W	w 11/77	W 11/57	ETB 11/17	11/00
19	E	e 12/65	E 12/45	ENQ 12/05	12/00
20	R	r 13/72	R 13/52	DC2 13/12	13/00
21	T	t 14/74	T 14/54	DC4 14/14	14/00
22	Y	y 15/79	Y 15/59	EM 15/19	15/00
23	U	u 16/75	U 16/55	NAK 16/15	16/00
24	I	i 17/69	I 17/49	HT 17/09	17/00
25	O	o 18/6F	O 18/4F	SI 18/0F	18/00
26	P	p 19/70	P 19/50	DLE 19/10	19/00
27	[{	[1A/5B	{ 1A/7B	ESC 1A/1B	**/**, 1A/00
28	] }	] 1B/5D	} 1B/7D	GS 1B/1D	**/**, 1B/00
29	＼ ¦	¥ 2B/5C	| 2B/7C	FS 2B/1C	**/**, 2B/00
30	CapsLock	**/**	**/**	**/**	**/**
31	A	a 1E/61	A 1E/41	SOH 1E/01	1E/00
32	S	s 1F/73	S 1F/53	DC3 1F/13	1F/00
33	D	d 20/64	D 20/44	EOT 20/04	20/00
34	F	f 21/66	F 21/46	ACK 21/06	21/00
35	G	g 22/67	G 22/47	BEL 22/07	22/00
36	H	h 23/68	H 23/48	BS 23/08	23/00
37	J	j 24/6A	J 24/4A	LF 24/0A	24/00
38	K	k 25/6B	K 25/4B	VT 25/0B	25/00
39	L	l 26/6C	L 26/4C	FF 26/0C	26/00
40	; :	; 27/3B	: 27/3A	**/**	**/**, 27/00
41	' "	' 28/27	" 28/22	**/**	**/**, 28/00
42	(キーなし)				
43	Enter	CR 1C/0D	CR 1C/0D	LF 1C/0A	**/**, 1C/00
44	Shift	**/**	**/**	**/**	**/**
45	(キーなし)				
46	Z	z 2C/7A	Z 2C/5A	SUB 2C/1A	2C/00
47	X	x 2D/78	X 2D/58	CAN 2D/18	2D/00
48	C	c 2E/63	C 2E/43	ETX 2E/03	2E/00
49	V	v 2F/76	V 2F/56	SYN 2F/16	2F/00
50	B	b 30/62	B 30/42	STX 30/02	30/00
51	N	n 31/6E	N 31/4E	SO 31/0E	31/00
52	M	m 32/6D	M 32/4D	CR 32/0D	32/00
53	, <	, 33/2C	< 33/3C	**/**	**/**, 33/00
54	. >	. 34/2E	> 34/3E	**/**	**/**, 34/00
55	／ ?	／ 35/2F	? 35/3F	**/**	**/**, 35/00
56	(キーなし)				
57	Shift	**/**	**/**	**/**	**/**
58	Ctrl	**/**	**/**	**/**	**/**
59	(キーなし)				
60	Alt	**/**	**/**	**/**	**/**
61	Space	SP 39/20	SP 39/20	SP 39/20	SP 39/20
62	Alt	**/**	**/**	**/**	**/**
63	(キーなし)				
64	Ctrl	**/**	**/**	**/**	**/**
65	(キーなし)				
66	(キーなし)				
67	(キーなし)				

IBM U.S. English キーボード (続き)

キー番号 刻印	下段	上段 (Shift+)	Ctrl+	Alt+
68 (キーなし)				
69 (キーなし)				
70 (キーなし)				
71 (キーなし)				
72 (キーなし)				
73 (キーなし)				
74 (キーなし)				
75 Insert	52/00, 52/E0	52/00, 52/E0	**/**, 92/E0	**/**, A2/00
76 Delete	53/00, 53/E0	53/00, 53/E0	**/**, 93/E0	**/**, A3/00
77 (キーなし)				
78 (キーなし)				
79 ←	4B/00, 4B/E0	4B/00, 4B/E0	73/00, 73/E0	**/**, 9B/00
80 Home	47/00, 47/E0	47/00, 47/E0	77/00, 77/E0	**/**, 97/00
81 End	4F/00, 4F/E0	4F/00, 4F/E0	75/00, 75/E0	**/**, 9F/00
82 (キーなし)				
83 ↑	48/00, 48/E0	48/00, 48/E0	**/**, 8D/E0	**/**, 98/00
84 ↓	50/00, 50/E0	50/00, 50/E0	**/**, 91/E0	**/**, A0/00
85 PageUp	49/00, 49/E0	49/00, 49/E0	84/00, 84/E0	**/**, 99/00
86 PageDown	51/00, 51/E0	51/00, 51/E0	76/00, 76/E0	**/**, A1/00
87 (キーなし)				
88 (キーなし)				
89 →	4D/00, 4D/E0	4D/00, 4D/E0	74/00, 74/E0	**/**, 9D/00
90 NumLock	**/**	**/**	**/**	**/**
91 Home 7	47/00	7 47/37	77/00	Code Input
92 ← 4	4B/00	4 4B/34	73/00	Code Input
93 End 1	4F/00	1 4F/31	75/00	Code Input
94 (キーなし)				
95 ／	／ 35/2F, E0/2F	／ 35/2F, E0/2F	**/**, 95/00	**/**, A4/00
96 ↑ 8	48/00	8 48/38	**/**, 8D/00	Code Input
97 5	**/**, 4C/00	5 4C/35	**/**, 8F/00	Code Input
98 ↓ 2	50/00	2 50/32	**/**, 91/00	Code Input
99 Ins 0	52/00	0 52/30	**/**, 92/00	Code Input
100 *	* 37/2A	* 37/2A	**/**, 96/00	**/**, 37/00
101 PgUp 9	49/00	9 49/39	84/00	Code Input
102 → 6	4D/00	6 4D/36	74/00	Code Input
103 PgDn 3	51/00	3 51/33	76/00	Code Input
104 Del .	53/00	. 53/2E	**/**, 93/00	**/**
105 −	− 4A/2D	− 4A/2D	**/**, 8E/00	**/**, 4A/00
106 +	+ 4E/2B	+ 4E/2B	**/**, 90/00	**/**, 4E/00
107 (キーなし)				
108 Enter	CR 1C/0D, E0/0D	CR 1C/0D, E0/0D	LF 1C/0A, E0/0A	**/**, A6/00
109 (キーなし)				
110 Esc	ESC 01/1B	ESC 01/1B	ESC 01/1B	**/**, 01/00
111 (キーなし)				
112 F1	3B/00	54/00	5E/00	68/00
113 F2	3C/00	55/00	5F/00	69/00
114 F3	3D/00	56/00	60/00	6A/00
115 F4	3E/00	57/00	61/00	6B/00
116 F5	3F/00	58/00	62/00	6C/00
117 F6	40/00	59/00	63/00	6D/00
118 F7	41/00	5A/00	64/00	6E/00
119 F8	42/00	5B/00	65/00	6F/00
120 F9	43/00	5C/00	66/00	70/00
121 F10	44/00	5D/00	67/00	71/00
122 F11	**/**, 85/00	**/**, 87/00	**/**, 89/00	**/**, 8B/00
123 F12	**/**, 86/00	**/**, 88/00	**/**, 8A/00	**/**, 8C/00
124 PrintScreen	**/**	**/**	72/00	**/**
125 ScrollLock	**/**	**/**	**/**	**/**
126 Pause	**/**	**/**	00/00	**/**
127 (キーなし)				
128 (キーなし)				
129 (キーなし)				
130 (キーなし)				
131 (キーなし)				
132 (キーなし)				
133 (キーなし)				

AX キーボード

キー番号	刻印	下段	上段 (Shift+)	Ctrl+	Alt+
1	Esc	ESC 01/1B	ESC 01/1B	ESC 01/1B	**/**, 01/00
2	1 ! ヌ	1 02/31	! 02/21	**/**	78/00
3	2 @ フ	2 03/32	@ 03/40	NUL 03/00	79/00
4	3 # ア ァ	3 04/33	# 04/23	**/**	7A/00
5	4 $ ウ ゥ	4 05/34	$ 05/24	**/**	7B/00
6	5 % エ ェ	5 06/35	% 06/25	**/**	7C/00
7	6 ^ オ ォ	6 07/36	^ 07/5E	RS 07/1E	7D/00
8	7 & ヤ ャ	7 08/37	& 08/26	**/**	7E/00
9	8 * ユ ュ	8 09/38	* 09/2A	**/**	7F/00
10	9 (ヨ ョ	9 0A/39	(0A/28	**/**	80/00
11	0) ワ ヲ	0 0B/30	) 0B/29	**/**	81/00
12	- _ ホ	- 0C/2D	_ 0C/5F	US 0C/1F	82/00
13	= + ヘ	= 0D/3D	+ 0D/2B	**/**	83/00
14	¥ | ー	¥ 2B/5C	| 2B/7C	FS 2B/1C	**/**, 2B/00
15	BackSpace	BS 0E/08	BS 0E/08	DEL 0E/7F	**/**, 0E/00
16	Tab Backtab	HT 0F/09	0F/00	**/**, 94/00	**/**, A5/00
17	Q タ	q 10/71	Q 10/51	DC1 10/11	10/00
18	W テ	w 11/77	W 11/57	ETB 11/17	11/00
19	E イ ィ	e 12/65	E 12/45	ENQ 12/05	12/00
20	R ス	r 13/72	R 13/52	DC2 13/12	13/00
21	T カ	t 14/74	T 14/54	DC4 14/14	14/00
22	Y ン	y 15/79	Y 15/59	EM 15/19	15/00
23	U ナ	u 16/75	U 16/55	NAK 16/15	16/00
24	I ニ	i 17/69	I 17/49	HT 17/09	17/00
25	O ラ	o 18/6F	O 18/4F	SI 18/0F	18/00
26	P セ	p 19/70	P 19/50	DLE 19/10	19/00
27	[{ ゛	[1A/5B	{ 1A/7B	ESC 1A/1B	**/**, 1A/00
28	] } ゜ 「	] 1B/5D	} 1B/7D	GS 1B/1D	**/**, 1B/00
29	(キーなし)				
30	Ctrl	**/**	**/**	**/**	**/**
31	A チ	a 1E/61	A 1E/41	SOH 1E/01	1E/00
32	S ト	s 1F/73	S 1F/53	DC3 1F/13	1F/00
33	D シ	d 20/64	D 20/44	EOT 20/04	20/00
34	F ハ	f 21/66	F 21/46	ACK 21/06	21/00
35	G キ	g 22/67	G 22/47	BEL 22/07	22/00
36	H ク	h 23/68	H 23/48	BS 23/08	23/00
37	J マ	j 24/6A	J 24/4A	LF 24/0A	24/00
38	K ノ	k 25/6B	K 25/4B	VT 25/0B	25/00
39	L リ	l 26/6C	L 26/4C	FF 26/0C	26/00
40	; : レ	; 27/3B	: 27/3A	**/**	**/**, 27/00
41	' " ケ	' 28/27	" 28/22	**/**	**/**, 28/00
42	` ~ ム 」	` 29/60	‾ 29/7E	**/**	**/**, 29/00
43	Enter	CR 1C/0D	CR 1C/0D	LF 1C/0A	**/**, 1C/00
44	Shift	**/**	**/**	**/**	**/**
45	(キーなし)				
46	Z ツ ッ	z 2C/7A	Z 2C/5A	SUB 2C/1A	2C/00
47	X サ	x 2D/78	X 2D/58	CAN 2D/18	2D/00
48	C ソ	c 2E/63	C 2E/43	ETX 2E/03	2E/00
49	V ヒ	v 2F/76	V 2F/56	SYN 2F/16	2F/00
50	B コ	b 30/62	B 30/42	STX 30/02	30/00
51	N ミ	n 31/6E	N 31/4E	SO 31/0E	31/00
52	M モ	m 32/6D	M 32/4D	CR 32/0D	32/00
53	, < ネ 、	, 33/2C	< 33/3C	**/**	**/**, 33/00
54	. > ル 。	. 34/2E	> 34/3E	**/**	**/**, 34/00
55	/ ? メ ・	/ 35/2F	? 35/3F	**/**	**/**, 35/00
56	＼ | ロ	¥ 56/5C	| 56/7C	FS 56/1C	**/**, 56/00
57	Shift	**/**	**/**	**/**	**/**
58	CapsLock	**/**	**/**	**/**	**/**
59	(キーなし)				
60	Alt	**/**	**/**	**/**	**/**
61	Space	SP 39/20	SP 39/20	SP 39/20	SP 39/20
62	漢字	3A/00	3A/00	**/**	**/**
63	(キーなし)				
64	英数カナ	**/**	**/**	**/**	**/**
65	(キーなし)				
66	(キーなし)				
67	(キーなし)				

AX キーボード (続き)

キー番号 刻印	下段	上段 (Shift+)	Ctrl+	Alt+
68 (キーなし)				
69 (キーなし)				
70 無変換	AB/00	AC/00	AD/00	AE/00
71 変換	A7/00	A8/00	A9/00	AA/00
72 (キーなし)				
73 (キーなし)				
74 (キーなし)				
75 Insert	52/00, 52/E0	52/00, 52/E0	**/**, 92/E0	**/**, A2/00
76 Delete	53/00, 53/E0	53/00, 53/E0	**/**, 93/E0	**/**, A3/00
77 (キーなし)				
78 (キーなし)				
79 ←	4B/00, 4B/E0	4B/00, 4B/E0	73/00, 73/E0	**/**, 9B/00
80 Home	47/00, 47/E0	47/00, 47/E0	77/00, 77/E0	**/**, 97/00
81 End	4F/00, 4F/E0	4F/00, 4F/E0	75/00, 75/E0	**/**, 9F/00
82 (キーなし)				
83 ↑	48/00, 48/E0	48/00, 48/E0	**/**, 8D/E0	**/**, 98/00
84 ↓	50/00, 50/E0	50/00, 50/E0	**/**, 91/E0	**/**, A0/00
85 PageUp	49/00, 49/E0	49/00, 49/E0	84/00, 84/E0	**/**, 99/00
86 PageDown	51/00, 51/E0	51/00, 51/E0	76/00, 76/E0	**/**, A1/00
87 (キーなし)				
88 (キーなし)				
89 →	4D/00, 4D/E0	4D/00, 4D/E0	74/00, 74/E0	**/**, 9D/00
90 NumLock	**/**	**/**	**/**	**/**
91 Home 7	47/00	7 47/37	77/00	Code Input
92 ← 4	4B/00	4 4B/34	73/00	Code Input
93 End 1	4F/00	1 4F/31	75/00	Code Input
94 (キーなし)				
95 ／	／ 35/2F, E0/2F	／ 35/2F, E0/2F	**/**, 95/00	**/**, A4/00
96 ↑ 8	48/00	8 48/38	**/**, 8D/00	Code Input
97 5	**/**, 4C/00	5 4C/35	**/**, 8F/00	Code Input
98 ↓ 2	50/00	2 50/32	**/**, 91/00	Code Input
99 Ins 0	52/00	0 52/30	**/**, 92/00	Code Input
100 *	* 37/2A	* 37/2A	**/**, 96/00	**/**, 37/00
101 PgUp 9	49/00	9 49/39	84/00	Code Input
102 → 6	4D/00	6 4D/36	74/00	Code Input
103 PgDn 3	51/00	3 51/33	76/00	Code Input
104 Del .	53/00	. 53/2E	**/**, 93/00	**/**
105 —	— 4A/2D	— 4A/2D	**/**, 8E/00	**/**, 4A/00
106 +	+ 4E/2B	+ 4E/2B	**/**, 90/00	**/**, 4E/00
107 (キーなし)				
108 Enter	CR 1C/0D, E0/0D	CR 1C/0D, E0/0D	LF 1C/0A, E0/0A	**/**, A6/00
109 (キーなし)				
110 AX	D2/00	D3/00	D4/00	D5/00
111 (キーなし)				
112 F1	3B/00	54/00	5E/00	68/00
113 F2	3C/00	55/00	5F/00	69/00
114 F3	3D/00	56/00	60/00	6A/00
115 F4	3E/00	57/00	61/00	6B/00
116 F5	3F/00	58/00	62/00	6C/00
117 F6	40/00	59/00	63/00	6D/00
118 F7	41/00	5A/00	64/00	6E/00
119 F8	42/00	5B/00	65/00	6F/00
120 F9	43/00	5C/00	66/00	70/00
121 F10	44/00	5D/00	67/00	71/00
122 F11	**/**, 85/00	**/**, 87/00	**/**, 89/00	**/**, 8B/00
123 F12	**/**, 86/00	**/**, 88/00	**/**, 8A/00	**/**, 8C/00
124 PrintScreen	**/**	**/**	72/00	**/**
125 ScrollLock	**/**	**/**	**/**	**/**
126 Pause	**/**	**/**	00/00	**/**
127 (キーなし)				
128 (キーなし)				
129 (キーなし)				
130 (キーなし)				
131 (キーなし)				
132 (キーなし)				
133 (キーなし)				

東芝 J-3100 キーボード

キー番号 刻印	下段	上段 (Shift+)	Ctrl+	Alt+
1 Esc	ESC 01/1B	ESC 01/1B	ESC 01/1B	**/**, 01/00
2 1 ! ぬ	1 02/31	! 02/21	**/**	78/00
3 2 @ ふ	2 03/32	@ 03/40	NUL 03/00	79/00
4 3 # あ ぁ	3 04/33	# 04/23	**/**	7A/00
5 4 $ う ぅ	4 05/34	$ 05/24	**/**	7B/00
6 5 % え ぇ	5 06/35	% 06/25	**/**	7C/00
7 6 ^ お ぉ	6 07/36	^ 07/5E	RS 07/1E	7D/00
8 7 & や ゃ	7 08/37	& 08/26	**/**	7E/00
9 8 * ゆ ゅ	8 09/38	* 09/2A	**/**	7F/00
10 9 (よ ょ	9 0A/39	(0A/28	**/**	80/00
11 0) わ	0 0B/30	) 0B/29	**/**	81/00
12 — _ ほ	— 0C/2D	_ 0C/5F	US 0C/1F	82/00
13 = + へ	= 0D/3D	+ 0D/2B	**/**	83/00
14 ¥ | ー	¥ 55/5C	| 55/7C	FS 55/1C	**/**, 2B/00
15 Backspace	BS 0E/08	BS 0E/08	DEL 0E/7F	**/**, 0E/00
16 Tab	HT 0F/09	0F/00	**/**, 94/00	**/**, A5/00
17 Q た	q 10/71	Q 10/51	DC1 10/11	10/00
18 W て	w 11/77	W 11/57	ETB 11/17	11/00
19 E い ぃ	e 12/65	E 12/45	ENQ 12/05	12/00
20 R す	r 13/72	R 13/52	DC2 13/12	13/00
21 T か	t 14/74	T 14/54	DC4 14/14	14/00
22 Y ん	y 15/79	Y 15/59	EM 15/19	15/00
23 U な	u 16/75	U 16/55	NAK 16/15	16/00
24 I に	i 17/69	I 17/49	HT 17/09	17/00
25 O ら	o 18/6F	O 18/4F	SI 18/0F	18/00
26 P せ	p 19/70	P 19/50	DLE 19/10	19/00
27 [{ ゛	[1A/5B	{ 1A/7B	GS 1A/1B	**/**, 1A/00
28] } ゜	] 1B/5D	} 1B/7D	FS 1B/1D	**/**, 1B/00
29 (キーなし)				
30 Ctrl	**/**	**/**	**/**	**/**
31 A ち	a 1E/61	A 1E/41	SOH 1E/01	1E/00
32 S と	s 1F/73	S 1F/53	DC3 1F/13	1F/00
33 D し	d 20/64	D 20/44	EOT 20/04	20/00
34 F は	f 21/66	F 21/46	ACK 21/06	21/00
35 G き	g 22/67	G 22/47	BEL 22/07	22/00
36 H く	h 23/68	H 23/48	BS 23/08	23/00
37 J ま	j 24/6A	J 24/4A	LF 24/0A	24/00
38 K の	k 25/6B	K 25/4B	VT 25/0B	25/00
39 L り	l 26/6C	L 26/4C	FF 26/0C	26/00
40 ; : れ	; 27/3B	: 27/3A	**/**	**/**, 27/00
41 ' " け	' 28/27	" 28/22	**/**	**/**, 28/00
42 ` ~ む 」	` 29/60	~ 29/7E	**/**	**/**, 29/00
43 Enter	CR 1C/0D	CR 1C/0D	LF 1C/0A	**/**, 1C/00
44 Shift	**/**	**/**	**/**	**/**
45 (キーなし)				
46 Z つ っ	z 2C/7A	Z 2C/5A	SUB 2C/1A	2C/00
47 X さ	x 2D/78	X 2D/58	CAN 2D/18	2D/00
48 C そ	c 2E/63	C 2E/43	ETX 2E/03	2E/00
49 V ひ	v 2F/76	V 2F/56	SYN 2F/16	2F/00
50 B こ	b 30/62	B 30/42	STX 30/02	30/00
51 N み	n 31/6E	N 31/4E	SO 31/0E	31/00
52 M も	m 32/6D	M 32/4D	CR 32/0D	32/00
53 , < ね 、	, 33/2C	< 33/3C	**/**	**/**, 33/00
54 . > る 。	. 34/2E	> 34/3E	**/**	**/**, 34/00
55 / ? め ・	/ 35/2F	? 35/3F	**/**	**/**, 35/00
56 ＼ ¦ ろ	¥ 2B/5C	| 2B/7C	FS 2B/1C	2B/00
57 Shift	**/**	**/**	**/**	**/**
58 CapsLock	**/**	**/**	**/**	**/**
59 (キーなし)				
60 Alt	**/**	**/**	**/**	**/**
61 Space	SP 39/20	SP 39/20	SP 39/20	SP 39/20
62 漢字	3A/00	A8/00	AA/00	B5/00
63 (キーなし)				
64 カナ	B6/00	B3/00	B6/00	B9/00
65 (キーなし)				
66 (キーなし)				
67 (キーなし)				

東芝 J-3100 キーボード (続き)

キー番号 刻印	下段	上段 (Shift+)	Ctrl+	Alt+
68 (キーなし)				
69 (キーなし)				
70 (キーなし)				
71 (キーなし)				
72 (キーなし)				
73 (キーなし)				
74 (キーなし)				
75 Insert	52/00, 52/E0	52/00, 52/E0	**/**, 92/E0	**/**, A2/00
76 Delete	53/00, 53/E0	53/00, 53/E0	**/**, 93/E0	**/**, A3/00
77 (キーなし)				
78 (キーなし)				
79 ←	4B/00, 4B/E0	4B/00, 4B/E0	73/00, 73/E0	**/**, 9B/00
80 Home	47/00, 47/E0	47/00, 47/E0	77/00, 77/E0	**/**, 97/00
81 End	4F/00, 4F/E0	4F/00, 4F/E0	75/00, 75/E0	**/**, 9F/00
82 (キーなし)				
83 ↑	48/00, 48/E0	48/00, 48/E0	**/**, 8D/E0	**/**, 98/00
84 ↓	50/00, 50/E0	50/00, 50/E0	**/**, 91/E0	**/**, A0/00
85 PageUp	49/00, 49/E0	49/00, 49/E0	84/00, 84/E0	**/**, 99/00
86 PageDown	51/00, 51/E0	51/00, 51/E0	76/00, 76/E0	**/**, A1/00
87 (キーなし)				
88 (キーなし)				
89 →	4D/00, 4D/E0	4D/00, 4D/E0	74/00, 74/E0	**/**, 9D/00
90 NumLock	**/**	**/**	**/**	**/**
91 Home 7	47/00	7 47/37	77/00	Code Input
92 ← 4	4B/00	4 4B/34	73/00	Code Input
93 End 1	4F/00	1 4F/31	75/00	Code Input
94 (キーなし)				
95 ／	／ 35/2F, E0/2F	／ 35/2F, E0/2F	**/**, 95/00	**/**, A4/00
96 ↑ 8	48/00	8 48/38	**/**, 8D/00	Code Input
97 5	**/**, 4C/00	5 4C/35	**/**, 8F/00	Code Input
98 ↓ 2	50/00	2 50/32	**/**, 91/00	Code Input
99 Ins 0	52/00	0 52/30	**/**, 92/00	Code Input
100 *	* 37/2A	* 37/2A	**/**, 96/00	**/**, 37/00
101 PgUp 9	49/00	9 49/39	84/00	Code Input
102 → 6	4D/00	6 4D/36	74/00	Code Input
103 PgDn 3	51/00	3 51/33	76/00	Code Input
104 Del .	53/00	. 53/2E	**/**, 93/00	**/**
105 −	− 4A/2D	− 4A/2D	**/**, 8E/00	**/**, 4A/00
106 +	+ 4E/2B	+ 4E/2B	**/**, 90/00	**/**, 4E/00
107 (キーなし)				
108 Enter	CR 1C/0D, E0/0D	CR 1C/0D, E0/0D	LF 1C/0A, E0/0A	**/**, A6/00
109 (キーなし)				
110	AF/00	D3/00	B2/00	B2/00
111 (キーなし)				
112 F1	3B/00	54/00	5E/00	68/00
113 F2	3C/00	55/00	5F/00	69/00
114 F3	3D/00	56/00	60/00	6A/00
115 F4	3E/00	57/00	61/00	6B/00
116 F5	3F/00	58/00	62/00	6C/00
117 F6	40/00	59/00	63/00	6D/00
118 F7	41/00	5A/00	64/00	6E/00
119 F8	42/00	5B/00	65/00	6F/00
120 F9	43/00	5C/00	66/00	70/00
121 F10	44/00	5D/00	67/00	71/00
122 F11	**/**, 85/00	**/**, 87/00	**/**, 89/00	**/**, 8B/00
123 F12	**/**, 86/00	**/**, 88/00	**/**, 8A/00	**/**, 8C/00
124 PrintScreen	**/**	**/**	72/00	**/**
125 ScrollLock	**/**	**/**	**/**	**/**
126 Pause	**/**	**/**	00/00	**/**
127 (キーなし)				
128 (キーなし)				
129 (キーなし)				
130 (キーなし)				
131 (キーなし)				
132 (キーなし)				
133 (キーなし)				

プリンター・データストリーム

アプリケーションからの出力は、文字コードとしてシフトJISを使います。コントロールコードは、ESC/P J84とします。ESC/Pの詳細は“EPSON ESC/P (TM) リファレンス・マニュアル 第2版”を参照してください。

ESC/PプリンターはJIS16進コード(JIS区点コードの16進表現に2020Hを加えた16進4桁のコード)を使います。アプリケーションの出力のシフトJISからJIS16進コードへの変換はプリンター・デバイス・ドライバーで行います。PC内部コードとしてのシフトJISは、メーカーにより若干の違いがありますが、プリンター・デバイス・ドライバー内でシフトJISからJIS16進コードへの変換時に処理されます。以下にデバイス・ドライバーでサポートする機能をあげます。

- 内部コードのシフトJISからプリンターのJIS16進コードへの変換を行ないます。コードの並び順が本体（たとえばJIS78）とプリンター（たとえばJIS83）で異なる場合は、デバイス・ドライバー内で対応する文字に自動変換することもできます。
- 各社選定文字は、ESC/Pプリンターの対応するコードに変換してからコードを送るか、外字定義機能を使ってイメージの転送、印刷を行います。その際、外字定義エリア (EC40H-EC9EH)をバッファーとして使用します。
- ユーザー定義文字は外字定義機能を使ってイメージの転送、印刷を行います。その際、外字定義エリア (EC40H-EC9EH) をバッファーとして使用します。

つまり、アプリケーションからはPC内部コードをそのまま“INT 17H”などを通じてはきだせばよいわけです。プリンター制御用にESC/Pコントロール・コードも混在できます。

JIS16進コードを使うこともできます。プリンター・ドライバーに特別なインターフェースは必要ありません。JIS16進コードは7FH以下なのでシフトJIS-JIS16進変換の対象にはならず、プリンターへパススルーされます。

ESC/Pコントロール・コードのFSD(半角縦書き2文字指定)などのように、コントロール・コード内で2バイト文字を指定する場合は、JIS16進コードを使います。この場合、このコントロール・コードに先立って漢字モード設定を行わなければなりません。

ディスケット・メディア・フォーマット

3.5インチ720KB、3.5インチ1.44MBを基本とします。

1.2MBはオプションで、機種によってサポートの有無、およびサポート方式の差があります。

アプリケーション供給メディアとしては、720KBか1.44MBのどちらかを使用してください。1.2MBを供給メディアとはしないでください。

文字コード体系（シフトJIS）

DOS/Vは、日本語環境と英語環境をシステムの再始動なしに連続的に切り換えて使用することができます。また再始動により切り換える英語モードもあります。

英語環境、または英語モードでは、文字セットとして1バイトコード体系のIBM PCコードを使います。“OADGハードウェア・インターフェース技術解説編”の『文字とキー・ストローク』の章を参照してください。

日本語環境ではコード体系としてシフトJISを使います。以下にOADGアプリケーションが使用可能なDOS/Vの2バイト・コード体系について説明します。

シフトJIS文字体系

OADGの文字コード体系はシフトJISを基本とし、以下のように定めます。2バイト文字はメーカーによる差が若干ありますので、この違いによらないようアプリケーションを作ってください。

DBCSベクター

文字列中のあるバイトが、1バイト文字か2バイト文字の1バイト目かを区別するにはDBCSベクターを使います。DBCSベクターは、2バイト文字の1バイト目のコードの範囲を示します。OADG DOS/VのDBCSベクターは次のとおりです。

81H-9FH
E0H-FCH

この値は日本用には固定ですが、他言語のサポートのため、DOSファンクション・コール“INT 21H AX=6300H: DBCSベクター情報の取得”または“INT 21H AH=65H: 拡張国別情報の取得”で返される値を参照するようプログラミングしてください。

ただし、文字列中の任意のバイトが2バイト文字の2バイト目であるかどうかはDBCSベクターからは判別できません。文字列の初めから調べるか、または判別できるまで文字列の先頭方向にさかのぼって検査してください。たとえば、文字列....8F 40 91 92 93 94...(16進) の94は2バイト文字の1バイト目ではないので、ここまでさかのぼれば判別できます。

1バイト文字セット

DBCSベクターの範囲外のコードは、1バイトとして扱います。表を参照してください。

1バイト文字セット

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		┼		0	@	P	`	p				ー	タ	ミ		
1	┌		!	1	A	Q	a	q			。	ア	チ	ム		
2	┐	↕	"	2	B	R	b	r			「	イ	ツ	メ		
3	└		#	3	C	S	c	s			」	ウ	テ	モ		
4	┘	█	$	4	D	T	d	t			、	エ	ト	ヤ		
5	│	┴	%	5	E	U	e	u			・	オ	ナ	ユ		
6	─	┬	&	6	F	V	f	v			ヲ	カ	ニ	ヨ		
7	↓	┤	'	7	G	W	g	w			ァ	キ	ヌ	ラ		
8			(	8	H	X	h	x			ィ	ク	ネ	リ		
9	○	├	)	9	I	Y	i	y			ゥ	ケ	ノ	ル		
A		▓	*	:	J	Z	j	z			ェ	コ	ハ	レ		
B	⌛	↵	+	;	K	[	k	{			ォ	サ	ヒ	ロ		
C		↑	,	<	L	¥	l	¦			ャ	シ	フ	ワ		
D		|	-	=	M	]	m	}			ュ	ス	ヘ	ン		
E	■	→	.	>	N	^	n	‾			ョ	セ	ホ	゛		
F	☼	←	/	?	O	_	o				ッ	ソ	マ	゜		

00H-1FH, 7FH:

1バイト制御コードまたは1バイト文字。キーボードからの入力、ディスプレイのTTY出力、テキスト・ファイル、通信用などには制御コードとして働きます。一部にディスプレイ表示用の特殊文字を定義しています。ディスプレイ上で枠などを描くときに使います。プリンターへの出力は保証されません。

20H-7EH:

1バイトの英数字。

A1H-DFH:

1バイトのカタカナ。

80H, A0H, FDH, FEH, FFH:

1バイトの文字としてメーカー予約。米国、欧州版のDOSではこれらのコードはすべて文字として扱われています。アプリケーション作成時に、これらのコードを独自の制御コードとして使わないでください。米国などのデータでは文字列にこれらのバイトが含まれていることがあり、データ交換で予期せぬ結果を生じることがあります。これらのバイトは未定義の文字として、他の定義されている文字と同様に扱ってください。

2バイト文字セット

JIS X0208準拠のシフトJISです。メーカーによって以下の違いがあります。

ユーザーがデータとして入力した2バイト文字コードは、特定のハードウェアで表示できない文字であっても、そのまま保持してください。

パネル、メッセージ、ヘルプなど、アプリケーションが出力する文字としては、共通でない文字を使わないようにしてください。ハードウェアによっては出力されないことがあります。たとえば、『丸で囲まれた数字』を持たないメーカーもあります。これらの文字を使うアプリケーションは、文字をユーザーが選択できるようにするなどしてください。

1. 字形の違い
   フォントは、メーカーによって、またディスプレイ・アダプター、プリンターなど表示装置の解像度によっても異なります。OADGではフォントの字形は規定しません。フォントはシステム、ハードウェアに依存します。

2. メーカー選定文字の違い
   メーカー選定文字には次のようなものがあります。

   - IBM選定文字: 386文字 (FA40H-FC4BH)
   - AX、エプソン選定文字: 82文字 (8740H-879CHの一部)
   - 東芝選定文字: 206文字 (81ADH-859EHの一部)

   ユーザーがこれらの文字をデータとして入力した場合、そのデータを保持してください。アプリケーションの表示用などにはこれらの文字を使わないでください。

3. ユーザー外字域の大きさの違い
   ユーザー外字域はユーザーが定義した文字をDOSが管理するためのコード領域です。アプリケーションが独自の文字を定義する領域ではありません。文字の作成や、かな漢辞書への登録はDOS付属のユーティリティーなどを使って行います。

   ユーザー外字域はF040Hから始まります(付録B 漢字コード表、図中の*4)。これはJIS94区に相当するシフトJISコードのすぐ後です。JISの区の定義は94区までですが、便宜上95区などと呼ぶことがあります。この区点番号はコード入力などで使います。

   たとえば、IBMではF040H-F9FCH (95区から114区まで)の1880文字です。

4. JISの制定年度による違い
   JIS X0208は83年度改訂のときに、文字の追加だけでなく、文字コードを一部入れ替えました。これにより83年以前の仕様のシフトJISと83年仕様のシフトJISでは、新字と旧字が入れ替わることがあります。これらの入れ替えに依存しないようにアプリケーションを作ってください。メッセージやヘルプ情報の表示用などで新字と旧字が入れ替わってもかまわない場合は使用してもかまいません。

   - JIS83年のコードポイント入れ替えによる違い以下の文字は83年仕様で入れ替わった文字です。

文字	区点番号 78	83	文字	区点番号 78	83	付録，図中の	
--	----	----	--	----	----		
鯵	8245	1619	鰺	1619	8245	*a	*A
鴬	8284	1809	鶯	1809	8284	*b	*B
蛎	7358	1934	蠣	1934	7358	*c	*C
撹	5788	1941	攪	1941	5788	*d	*D
竃	6762	1986	竈	1986	6762	*e	*E
潅	6285	2035	灌	2035	6285	*f	*F
諌	7561	2050	諫	2050	7561	*g	*G
頚	8084	2359	頸	2359	8084	*h	*H
砿	6672	2560	礦	2560	6672	*i	*I
蕊	7302	2841	蘂	2841	7302	*j	*J
靭	8055	3157	靱	3157	8055	*k	*K
賎	7645	3308	賤	3308	7645	*l	*L
壷	5268	3659	壺	3659	5268	*m	*M

砺	6674	3755	礪	3755	6674	*n	*N
梼	5977	3778	檮	3778	5977	*o	*O
涛	6225	3783	濤	3783	6225	*p	*P
迩	7778	3886	邇	3886	7778	*q	*Q
蝿	7404	3972	蠅	3972	7404	*r	*R
桧	5956	4116	檜	4116	5956	*s	*S
侭	4854	4389	儘	4389	4854	*t	*T
薮	7314	4489	藪	4489	7314	*u	*U
篭	6838	4722	籠	4722	6838	*v	*V
--	----	----	--	----	----		
尭		2238	堯	2238	8401	*w	*W
槙		4374	槇	4374	8402	*x	*X
遥		4558	遙	4558	8403	*y	*Y
瑶		6486	瑤	6486	8404	*z	*Z

- JIS83年とIBM選定文字の重複による違い
  次の2文字は83年の定義以前からIBMが選定文字として使っていたもので、
  IBMの場合コードポイントが異なります。

  と　　(2文字)　　　付録B、図中の*1, *2

- 90年の2文字の追加
  90年仕様で2文字追加になりましたが、インプリメントはメーカー依存です。
  (付録B、図中の*3)

付録Bの漢字コード表はこれらの考慮すべき文字を示しています。

米国アプリケーション・プログラムの互換性

DOS/Vは米国のDOS/BIOSを拡張したものですので、米国で販売されているアプリケーション・ソフトウェアは、BIOS割り込みを使用していればそのまま起動することができます。ただし、次にあげるようなハードウェア、文字セット上の制限がありますので、運用上ご注意ください。

- テキスト・モードで、テキスト・バッファーを直接アクセスしているソフトウェアは画面が表示されません。これはDOS/Vが2バイト文字を表示するために、グラフィック・モードでテキスト・モードをエミュレートしているためです。テキスト・バッファーには書き込まれていますが、表示用のグラフィック・モードのバッファーにフォントが展開されないためです。

  この場合、英語環境または英語モードに切り換えると正常に表示されます。ただし、英語環境、英語モードではコードページが437のため日本語を表示、入力することはできません。

  BIOS割り込みを使っているソフトウェアにはこの問題はありません。

- テキスト・アプリケーションで枠などを表示するために80H以上のコードを使っている場合、コードページの違いにより、半角カタカナなどで表示されることがあります。

  表示用だけならこのままでも使えます。入力に使う場合は、英語環境、英語モードに切り換えて使います。ただし、この場合、日本語の表示、入力はできません。

- キーボードからの入力としてINT 16Hの文字コードを使っていない場合、またはINT 09Hを使っている場合はキーボード上の文字が正しく入力されないことがあります。この問題は特にJIS配列のキーボードで起こります。JIS配列とASCII配列の違い(数字キーの上段、アルファベットとEnterキーの間の特殊記号など)によるものです。たとえば、JIS配列の2の上段の“"”を押したつもりでもASCII配列と同じ“@”が表紙されることになります。

  このようなキーを頻繁に使わないのであれば、ASCII配列で定義された位置のキーを押すか、Alt+キーパッド数値キーのコード入力でタイプできます。

  ASCII配列のキーボードではこの問題はありません。またアプリケーションがINT 16Hの文字コードを参照していれば正しく働きます。

- 豊富な印刷機能を備えた米国のアプリケーションの多くはプリンター・ドライバーを持っています。使用中のプリンターをサポートしていないアプリケーションでは正しく印刷できないことがあります。

- 通常米国のアプリケーションは文字を1バイトと認識しています。2バイト文字の処理を明言していない場合、文字の検索、変換、ソートなどの機能は2バイト文字のデータに対して正しく働かないことがあります。たとえば、文字“A”(41H)を文字“a”(61H)に変える変換は2バイト文字“帰” (8B41H)を“蟻”(8B61H)に変えてしまうことがあります。

このようなアプリケーションで2バイト文字を含むデータを扱う場合、文字列の一括変換などは注意して行ってください。1バイトの英数カナ文字のみであればこの問題は起こりません。

- カーソルをアプリケーションが制御している場合、2バイト文字の右半分、または中央におかれることがあります。これは2バイトで1文字であると認識していないためです。

  このカーソル位置で文字の入力、削除などを行いますと、文字化けが起こります。カーソルを2バイト文字の左半分、または文字と文字の間に移してから操作してください。

- 2バイト文字をファイル名やパス名に使っていると、2バイト文字の2バイト目をパスの区切りとして扱われることがあります。たとえば、“ソ”(835CH)の2バイト目が“¥”(5CH)として扱われることになります。

  米国のソフトウェアを多用する場合は、ファイル名、パス名を英数字に制限することをお奨めします。

米国アプリケーション・プログラムの移植

DOS/VのBIOSインターフェースは基本的にPS/2またはAT互換機のBIOSインターフェースに定義されているインターフェースを使用することができるようになっています。

ここでは、米国のアプリケーション・プログラムをDOS/V上で動作させせた場合に生ずる前記したいくつかの互換性に対する問題点を吸収し、2バイト文字セットが使用できるように移植する際の考慮事項について説明します。

ビデオ

ビデオ・モード: DOS/Vでは、次のビデオ・モードが使用できます。

- 03H - エミュレートCGA文字モード
- 11H - 2色カラー・グラフィック・モード
- 12H - 16色カラー・グラフィック・モード
- 72H - 16色カラー・グラフィック・モード
- 73H - エミュレート拡張CGA文字モード

ビデオ・モードの詳細は、『DOS/V BIOSインターフェース技術解説編』の“INT 10H, AH=00H ビデオ・モードの設定”を参照してください。

ハードウェアに直接アクセスせずに、DOS/BIOSインターフェースを使用するようなアプリケーション・プログラムに関しては、エミュレートCGA文字モード - 03Hは、PS/2またはPC/AT互換機BIOSの CGA文字モードと互換性があります。

2つのモードは、下表に示す項目を除いて等価です。

図　1-5. CGA文字モードとエミュレートCGA文字モードの比較		
比較項目	**CGA文字モード**	**エミュレートCGA文字モード**
テキスト・バッファー	B8000H-	疑似テキスト・バッファー（INT10H, AH=FEHで取得できます）
文字ジェネレーター・バッファー	最大2ブロック	1ブロック（擬似文字ジェネレーター・バッファー）
文字サイズ	8×8（VGAの場合、9×16）	8×19
文字の点滅	サポートあり	背景色の輝度で代用
カーソル	前景色で点滅	点滅しない（APAをFFHの値でXOR）
ページ数	4（VGAの場合、8）	1

ビデオ・モード73Hは、ビデオ・モード03Hの文字属性を拡張したものです。次の機能が追加されています。

- 下線
- 横罫線と縦罫線

ビデオ・モード11Hと12Hは2バイト文字セットの読み・書きを除いて完全にVGAのビデオ・モードと互換性があります。

ビデオ・モード72Hは、文字モードの表示サイズ80×25と同じで、グラフィック・モードでビデオ・モード03Hや73Hと同様の文字表示を行うのに便利です。

エミュレート文字モード:　この文字モードは、グラフィック・モードでエミュレートされるため、既存のアプリケーション・プログラムが直接にテキスト・バッファーをアクセスする場合、正しく作動しません。バッファーを直接アクセスするようなプログラムは、BIOSインターフェースのみを使用してアクセスするように修正する必要があります。修正のガイド・ラインを以下に示します。

1. 文字列の読み・書きに関してINT10H、AH=FEH/FFHを使用する

   この機能は、ビデオ・モード03Hのみに使用可能です。

   これらの機能は、次に示す方法によりプログラムの修正に費やす負荷を軽減するのに役立ちます。これらの機能を使用してアプリケーション・プログラムは疑似テキスト・バッファーを直接アクセスできます。これを実行するには、機能AH=FEH

により返されたバッファーをアクセスし、バッファー上の変更個所をディスプレイ上に表示しなければならない時点で‘AH=FFH画面表示の更新’機能を呼び出します。この機能は指定された文字列をビット・イメージに変換して画面上に表示します。

ハードウェアのテキスト・バッファーが存在する場合は、INT10H, AH=FEH/FFHは何ら機能しません。INT10H, AH=FEHを呼び出すときに、ハードウェア・ビデオ・バッファーのアドレス(B8000H)を指定し、返されたアドレス（このときはB8000Hのまま）のテキスト・バッファーに対して文字を書き込めば、それらがディスプレイ上に表示されます。その後に出すINT10H, AH=FFHは表示に影響を及ぼしません。

この様に作成されたアプリケーション・プログラムは、ハードウェアのテキスト・バッファーの存在いかんにかかわらず、正しく作動することができます。

2. 文字列の書き込みに関してINT10H, AX=130xHを使用する

   このインターフェースは、DOS/VとPS/2またはPC/AT互換機のBIOSとの間で共通のもので、現在のカーソル位置を移動しても、しなくても高速でどの位置（行と桁数）にでも文字列を書き込むことができます。制御文字はコマンドとして取り扱われることに注意してください。

3. 文字列の読み・書きに関するINT10H, AX=131xH/132xHを使用する

   このインターフェースは現在のカーソル位置を移動することなく、高速でどの位置（行と桁数）でも文字列の読み込みや文字列の書き込みが可能です。制御文字も通常の文字と同じ扱いを受けます。そのため、この処理は文字画面の更新と読み取りに関して使用されるテキスト・バッファーから、またはテキスト・バッファーへの‘MOVSW’命令に代えて使用することができます。

   ただし、INT10H, AX=130xHをできる限り、使用されることをお勧めします。

4. APAに直接アクセスしない

   アプリケーション・プログラムがAPAに直接アクセスして線などを描いた場合、テキストモードで画面イメージをコードと属性の組み合わせで保管、復元するような常駐プログラムがあると、描かれたイメージは復元できません。

コード・ページ

DOS/Vはコード・ページ932のみサポートします。コード・ページ932は、多くの英語アプリケーション・プログラムが使用しているコード・ページ437のすべての文字を処理できるようになっていないため、一部の図形文字に代わって半角カタカナが表示されたりします。これら一部のコードはB3H～DAHの範囲に割り当てられたもので、日本語の半角カタカナ文字と重複しています。

このような表示を防ぐために、１つの方法としてはコード・ページ437の図形文字をコード・ページ932の図形文字に変更します。これは英語から日本語へのメッセージ翻訳の一環として考えることもできます。

もう１つの方法は、半角カタカナ文字のコードを適用業務プログラムが使用しないのであれば、図形文字が表示できるように図形の文字イメージに変更します。
ただしこの方法は、日本語MS-WINDOWS V3.0のウィンドーDOS互換モード等、サポートされない場合があります。

キーボード

キーボードの走査コード

IBM 5576-A01, AX, 東芝J-3100などの日本語に対応したキーボードではポート60Hから読み取られるキーボードの走査コードは、IBM U.S. English (101キー)キーボードの走査コード・セット01Hのスーパーセットです。一部の走査コードが2バイト文字セットの拡張キー用に追加されています。これらのキーは、INT16Hインターフェースを介してアプリケーション・プログラムに提供される前に日本語入力サブシステムにより処理されます。アプリケーション・プログラムが、INT9H（ハードウェア・キーボード割り込み）の割り込みにより、またはINT15H, AH=4FH（キーボード・インターセプト）により生のキーボード走査コードを読み取る場合は、それらを正しく処理するか、無視してください。

さらに、DOS/VでサポートするキーボードにはASCII配列とJISCII配列との違いがあるため、生のキーボード走査コードを読み取るアプリケーション・プログラムは、キーの表記に注意を払う必要があります。

プリンター

プリンターのデータ・ストリーム: プリンターのデータ・ストリームは接続しているプリンター・タイプとの関連はありますが、BIOSインターフェースとの関係はありません。

PS/2またはPC/AT互換機用のアプリケーション・プログラムの多くは、IBM用のプロプリンターの制御コードを使用しています。したがって、INT17H, AH=00Hで文字のみを印刷する場合以外は、対応するそれぞれのプリンター制御コードに従って印刷の出力データを変更する必要があります。

2バイト文字セットの使用

2バイト文字セットを使用可能にするには、以下の事項について考慮する必要があります。

- 2バイト文字セットの文字は、キーボード入出力のBIOS(またはDOS)機能を連続して2度呼び出して読み取る必要があります。次に入力される文字が半角文字と決めてかからないようにしてください。
- ディスプレイ上には、2バイト文字セットの文字は、部分的に上書きして残さないようにしてください。文字の右側または左側が他の文字で上書きされた場合、その反対側も上書きするか、スペース文字でブランクにする必要があります。
- ディスプレイ上では、各文字行の最初の位置は半角文字か全角文字の1バイト目で始まり、各文字行の最後の位置は半角文字か全角文字の2バイト目で終わるようにしてください。

- 2バイト文字のコード範囲（DBCSベクター）を1バイト文字セットまたは2バイト文字セットのいずれかに決めつけるようなプログラムを作成しないでください。
  DBCSベクターはDOS機能呼び出しINT21H, AX=6300Hで取得することができます。これによって、プログラムは１バイト文字セット用と２バイト文字セット用のいずれでも動作することが可能になります。

付録A. DOS/V 5.0とDOS/V 6の相違点

DOS/V 5.0からDOS/V 6 では以下のBIOSコール及びファンクション・コールが変更/追加されています。詳細は、『DOS/V BIOSインターフェース 技術解説編』および『DOS/V 技術解説編』を参照してください。

OADG DOS/V 5.0からDOS/V 6 ではファンクション・コールの変更/追加はありません。

	ファンクション	OADG	MS-DOS 5.0/V	PC-DOS 6.3/V	MS-DOS 6.2/V
Int 10h					
1Dh	ディスプレイシステムラインの表示／消去／行数取込み	○	○	○	×
Int 16h					
13h	2バイト文字セットの状況モードの設定と読み出し	○	○	○	×
14h	シフト状況の表示と消去	○	○	○	×
Int 21h					
440Dh					
47h	HDアクセスフラグの設定	○	×	○	×(US)
67h	HDアクセスフラグの設定	○	×	○	×(US)

注: MS-DOSは米マイクロソフト社の登録商標です。

＜参考＞ DOS/V4.0 と DOS/V5.0 の相違点

INT 13h	AH=08h	BL	ディスケットの情報
INT 21h	AH=30h AH=33h AH=44h	AL=0Dh AL=10h AL=11h	DOSバージョン番号の取得 システム値の取得および設定 装置の入出力制御 装置の入出力制御 装置の入出力制御
INT 2Fh	AH=10h AX=1680h		多重割り込み 多重割り込み

付録B. 漢字コード表

□で囲った文字とコードについては本文の『文字コード体系』を参照してください。

シフト JIS	+0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
8140		、	。	，	．	・	：	；	？	！	゛	゜	´	｀	¨	＾	￣	＿	ヽ	ヾ	ゝ	ゞ	〃	仝	々	〆	〇	ー	―	‐	／	＼
8160	～	∥	｜	…	‥	‘	’	“	”	（	）	〔	〕	［	］	｛	｝	〈	〉	《	》	「	」	『	』	【	】	＋	－	±	×	
8180	÷	＝	≠	＜	＞	≦	≧	∞	∴	♂	♀	°	′	″	℃	￥	＄	￠	￡	％	＃	＆	＊	＠	§	☆	★	○	●	◎	◇	◆
81A0	□	■	△	▲	▽	▼	※	〒	→	←	↑	↓	〓												∈	∋	⊆	⊇	⊂	⊃	∪	∩
81C0									∧	∨	*1	⇒	⇔	∀	∃												∠	⊥	⌒	∂	∇	≡
81E0	≒	≪	≫	√	∽	∝	*2	∫	∬								Å	‰	♯	♭	♪	†	‡	¶					◯			
8240																	０	１	２	３	４	５	６	７	８	９						
8260	Ａ	Ｂ	Ｃ	Ｄ	Ｅ	Ｆ	Ｇ	Ｈ	Ｉ	Ｊ	Ｋ	Ｌ	Ｍ	Ｎ	Ｏ	Ｐ	Ｑ	Ｒ	Ｓ	Ｔ	Ｕ	Ｖ	Ｗ	Ｘ	Ｙ	Ｚ						
8280		ａ	ｂ	ｃ	ｄ	ｅ	ｆ	ｇ	ｈ	ｉ	ｊ	ｋ	ｌ	ｍ	ｎ	ｏ	ｐ	ｑ	ｒ	ｓ	ｔ	ｕ	ｖ	ｗ	ｘ	ｙ	ｚ					ぁ
82A0	あ	ぃ	い	ぅ	う	ぇ	え	ぉ	お	か	が	き	ぎ	く	ぐ	け	げ	こ	ご	さ	ざ	し	じ	す	ず	せ	ぜ	そ	ぞ	た	だ	ち
82C0	ぢ	っ	つ	づ	て	で	と	ど	な	に	ぬ	ね	の	は	ば	ぱ	ひ	び	ぴ	ふ	ぶ	ぷ	へ	べ	ぺ	ほ	ぼ	ぽ	ま	み	む	め
82E0	も	ゃ	や	ゅ	ゆ	ょ	よ	ら	り	る	れ	ろ	ゎ	わ	ゐ	ゑ	を	ん														
8340	ァ	ア	ィ	イ	ゥ	ウ	ェ	エ	ォ	オ	カ	ガ	キ	ギ	ク	グ	ケ	ゲ	コ	ゴ	サ	ザ	シ	ジ	ス	ズ	セ	ゼ	ソ	ゾ	タ	ダ
8360	チ	ヂ	ッ	ツ	ヅ	テ	デ	ト	ド	ナ	ニ	ヌ	ネ	ノ	ハ	バ	パ	ヒ	ビ	ピ	フ	ブ	プ	ヘ	ベ	ペ	ホ	ボ	ポ	マ	ミ	
8380	ム	メ	モ	ャ	ヤ	ュ	ユ	ョ	ヨ	ラ	リ	ル	レ	ロ	ヮ	ワ	ヰ	ヱ	ヲ	ン	ヴ	ヵ	ヶ									Α
83A0	Β	Γ	Δ	Ε	Ζ	Η	Θ	Ι	Κ	Λ	Μ	Ν	Ξ	Ο	Π	Ρ	Σ	Τ	Υ	Φ	Χ	Ψ	Ω									α
83C0	β	γ	δ	ε	ζ	η	θ	ι	κ	λ	μ	ν	ξ	ο	π	ρ	σ	τ	υ	φ	χ	ψ	ω									
83E0																																
8440	А	Б	В	Г	Д	Е	Ё	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю
8460	Я																а	б	в	г	д	е	ё	ж	з	и	й	к	л	м	н	
8480	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я														─
84A0	│	┌	┐	┘	└	├	┬	┤	┴	┼	━	┃	┏	┓	┛	┗	┣	┳	┫	┻	╋	┠	┯	┨	┷	┿	┝	┰	┥	┸	╂	
84C0																																
84E0																																
8540																																
8560																																
8580																																
85A0																																
85C0																																
85E0																																
8640																																
8660																																
8680																																
86A0																																
86C0																																
86E0																																
8740																																
8760																																
8780																																
87A0																																
87C0																																
87E0																																

*1:¬
*2:∵

シフトJIS +	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	
8840																																	
8860																																	
8880																																亜	
88A0	唖	娃	阿	哀	愛	挨	姶	逢	葵	茜	穐	悪	握	渥	旭	葦	芦	*A	梓	圧	斡	扱	宛	姐	虻	飴	絢	綾	鮎	或	粟	袷	*A:鰺/鯵
88C0	安	庵	按	暗	案	闇	鞍	杏	以	伊	位	依	偉	囲	夷	委	威	尉	惟	意	慰	易	椅	為	畏	異	移	維	緯	胃	萎	衣	
88E0	謂	違	遺	医	井	亥	域	育	郁	磯	一	壱	溢	逸	稲	茨	芋	鰯	允	印	咽	員	因	姻	引	飲	淫	胤	蔭				
8940	院	陰	隠	韻	吋	右	宇	烏	羽	迂	雨	卯	鵜	窺	丑	碓	臼	渦	嘘	唄	欝	蔚	鰻	姥	厩	浦	瓜	閏	噂	云	運	雲	
8960	荏	餌	叡	営	嬰	影	映	曳	栄	永	泳	洩	瑛	盈	穎	頴	英	衛	詠	鋭	液	疫	益	駅	悦	謁	越	閲	榎	厭	円		
8980	園	堰	奄	宴	延	怨	掩	援	沿	演	炎	焔	煙	燕	猿	縁	艶	苑	薗	遠	鉛	鴛	塩	於	汚	甥	凹	央	奥	往	応	押	
89A0	旺	横	欧	殴	王	翁	襖	*B	鴎	黄	岡	沖	荻	億	屋	憶	臆	桶	牡	乙	俺	卸	恩	温	穏	音	下	化	仮	何	伽	価	*B:鶯/鴬
89C0	佳	加	可	嘉	夏	嫁	家	寡	科	暇	果	架	歌	河	火	珂	禍	禾	稼	箇	花	苛	茄	荷	華	菓	蝦	課	嘩	貨	迦	過	
89E0	霞	蚊	俄	峨	我	牙	画	臥	芽	蛾	賀	雅	餓	駕	介	会	解	回	塊	壊	廻	快	怪	悔	恢	懐	戒	拐	改				
8A40	魁	晦	械	海	灰	界	皆	絵	芥	蟹	開	階	貝	凱	劾	外	咳	害	崖	慨	概	涯	碍	蓋	街	該	鎧	骸	浬	馨	蛙	垣	*C:蠣/蛎
8A60	柿	*C	鈎	劃	嚇	各	廓	拡	*D	格	核	殻	獲	確	穫	覚	角	赫	較	郭	閣	隔	革	学	岳	楽	額	顎	掛	笠	樫		*D:攪/撹
8A80	橿	梶	鰍	潟	割	喝	恰	括	活	渇	滑	葛	褐	轄	且	鰹	叶	椛	樺	鞄	株	兜	*E	蒲	釜	鎌	噛	鴨	栢	茅	萱	粥	*E:竈/竃
8AA0	刈	苅	瓦	乾	侃	冠	寒	刊	勘	勧	巻	喚	堪	姦	完	官	寛	干	幹	患	感	慣	憾	換	敢	柑	桓	棺	款	歓	汗	漢	
8AC0	澗	*F	環	甘	監	看	竿	管	簡	緩	缶	翰	肝	艦	莞	観	*G	貫	還	鑑	間	閑	関	陥	韓	館	舘	丸	含	岸	巌	玩	*F:灌/潅
8AE0	癌	眼	岩	翫	贋	雁	頑	顔	願	企	伎	危	喜	器	基	奇	嬉	寄	岐	希	幾	忌	揮	机	旗	既	期	棋	棄				*G:諫/諌
8B40	機	帰	毅	気	汽	畿	祈	季	稀	紀	徽	規	記	貴	起	軌	輝	飢	騎	鬼	亀	偽	儀	妓	宜	戯	技	擬	欺	犠	疑	祇	
8B60	義	蟻	誼	議	掬	菊	鞠	吉	吃	喫	桔	橘	詰	砧	杵	黍	却	客	脚	虐	逆	丘	久	仇	休	及	吸	宮	弓	急	救		
8B80	朽	求	汲	泣	灸	球	究	窮	笈	級	糾	給	旧	牛	去	居	巨	拒	拠	挙	渠	虚	許	距	鋸	漁	禦	魚	亨	享	京	供	
8BA0	侠	僑	兇	競	共	凶	協	匡	卿	叫	喬	境	峡	強	彊	怯	恐	恭	挟	教	橋	況	狂	狭	矯	胸	脅	興	蕎	郷	鏡	響	
8BC0	饗	驚	仰	凝	*W	暁	業	局	曲	極	玉	桐	粁	僅	勤	均	巾	錦	斤	欣	欽	琴	禁	禽	筋	緊	芹	菌	衿	襟	謹	近	*W:堯/尭
8BE0	金	吟	銀	九	倶	句	区	狗	玖	矩	苦	躯	駆	駈	駒	具	愚	虞	喰	空	偶	寓	遇	隅	串	櫛	釧	屑	屈				
8C40	掘	窟	沓	靴	轡	窪	熊	隈	粂	栗	繰	桑	鍬	勲	君	薫	訓	群	軍	郡	卦	袈	祁	係	傾	刑	兄	啓	圭	珪	型	契	
8C60	形	径	恵	慶	慧	憩	掲	携	敬	景	桂	渓	畦	稽	系	経	継	繋	罫	茎	荊	蛍	計	詣	警	軽	*H	鶏	芸	迎	鯨		*H:頸/頚
8C80	劇	戟	撃	激	隙	桁	傑	欠	決	潔	穴	結	血	訣	月	件	倹	倦	健	兼	券	剣	喧	圏	堅	嫌	建	憲	懸	拳	捲	検	
8CA0	権	牽	犬	献	研	硯	絹	県	肩	見	謙	賢	軒	遣	鍵	険	顕	験	鹸	元	原	厳	幻	弦	減	源	玄	現	絃	舷	言	諺	
8CC0	限	乎	個	古	呼	固	姑	孤	己	庫	弧	戸	故	枯	湖	狐	糊	袴	股	胡	菰	虎	誇	跨	鈷	雇	顧	鼓	五	互	伍	午	
8CE0	呉	吾	娯	後	御	悟	梧	檎	瑚	碁	語	誤	護	醐	乞	鯉	交	佼	侯	候	倖	光	公	功	効	勾	厚	口	向				
8D40	后	喉	坑	垢	好	孔	孝	宏	工	巧	巷	幸	広	庚	康	弘	恒	慌	抗	拘	控	攻	昂	晃	更	杭	校	梗	構	江	洪	浩	
8D60	港	溝	甲	皇	硬	稿	糠	紅	紘	絞	綱	耕	考	肯	肱	腔	膏	航	荒	行	衡	講	貢	購	郊	酵	鉱	*I	鋼	閤	降		*I:礦/砿
8D80	項	香	高	鴻	剛	劫	号	合	壕	拷	濠	豪	轟	麹	克	刻	告	国	穀	酷	鵠	黒	獄	漉	腰	甑	忽	惚	骨	狛	込	此	
8DA0	頃	今	困	坤	墾	婚	恨	懇	昏	昆	根	梱	混	痕	紺	艮	魂	些	佐	叉	唆	嵯	左	差	査	沙	瑳	砂	詐	鎖	裟	坐	
8DC0	座	挫	債	催	再	最	哉	塞	妻	宰	彩	才	採	栽	歳	済	災	采	犀	砕	砦	祭	斎	細	菜	裁	載	際	剤	在	材	罪	
8DE0	財	冴	坂	阪	堺	榊	肴	咲	崎	埼	碕	鷺	作	削	咋	搾	昨	朔	柵	窄	策	索	錯	桜	鮭	笹	匙	冊	刷				
8E40	察	拶	撮	擦	札	殺	薩	雑	皐	鯖	捌	錆	鮫	皿	晒	三	傘	参	山	惨	撒	散	桟	燦	珊	産	算	纂	蚕	讃	賛	酸	
8E60	餐	斬	暫	残	仕	仔	伺	使	刺	司	史	嗣	四	士	始	姉	姿	子	屍	市	師	志	思	指	支	孜	斯	施	旨	枝	止		
8E80	死	氏	獅	祉	私	糸	紙	紫	肢	脂	至	視	詞	詩	試	誌	諮	資	賜	雌	飼	歯	事	似	侍	児	字	寺	慈	持	時	次	
8EA0	滋	治	爾	璽	痔	磁	示	而	耳	自	蒔	辞	汐	鹿	式	識	鴫	竺	軸	宍	雫	七	叱	執	失	嫉	室	悉	湿	漆	疾	質	
8EC0	実	蔀	篠	偲	柴	芝	屡	*J	縞	舎	写	射	捨	赦	斜	煮	社	紗	者	謝	車	遮	蛇	邪	借	勺	尺	杓	灼	爵	酌	釈	*J:蘂/蕊
8EE0	錫	若	寂	弱	惹	主	取	守	手	朱	殊	狩	珠	種	腫	趣	酒	首	儒	受	呪	寿	授	樹	綬	需	囚	収	周				
8F40	宗	就	州	修	愁	拾	洲	秀	秋	終	繍	習	臭	舟	蒐	衆	襲	讐	蹴	輯	週	酋	酬	集	醜	什	住	充	十	従	戎	柔	
8F60	汁	渋	獣	縦	重	銃	叔	夙	宿	淑	祝	縮	粛	塾	熟	出	術	述	俊	峻	春	瞬	竣	舜	駿	准	循	旬	楯	殉	淳		
8F80	準	潤	盾	純	巡	遵	醇	順	処	初	所	暑	曙	渚	庶	緒	署	書	薯	藷	諸	助	叙	女	序	徐	恕	鋤	除	傷	償	勝	
8FA0	匠	升	召	哨	商	唱	嘗	奨	妾	娼	宵	将	小	少	尚	庄	床	廠	彰	承	抄	招	掌	捷	昇	昌	昭	晶	松	梢	樟	樵	
8FC0	沼	消	渉	湘	焼	焦	照	症	省	硝	礁	祥	称	章	笑	粧	紹	肖	菖	蒋	蕉	衝	裳	訟	証	詔	詳	象	賞	醤	鉦	鍾	
8FE0	鐘	障	鞘	上	丈	丞	乗	冗	剰	城	場	壌	嬢	常	情	擾	条	杖	浄	状	畳	穣	蒸	譲	醸	錠	嘱	埴	飾				

シフトJIS	+ 0 1 2 3 4 5 6 7 8 9 A B C D E F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F	
9040	拭植殖燭織職色触食蝕辱尻伸信侵唇娠寝審心慎振新晋森榛浸深申疹真神	
9060	秦紳臣芯薪親診身辛進針震人仁刃塵壬尋甚尽腎訊迅陣[*K]笥諏須酢図厨	*K:靱/靭
9080	逗吹垂帥推水炊睡粋翠衰遂酔錐錘随瑞髄崇嵩数枢趨雛据杉椙菅頗雀裾澄	
90A0	摺寸世瀬畝是凄制勢姓征性成政整星晴棲栖正清牲生盛精聖声製西誠誓請	
90C0	逝醒青静斉税脆隻席惜戚斥昔析石積籍績脊責赤跡蹟碩切拙接摂折設窃節	
90E0	説雪絶舌蝉仙先千占宣専尖川戦扇撰栓栴泉浅洗染潜煎煽旋穿箭線	
9140	繊羨腺舛船薦詮[*L]践選遷銭銑閃鮮前善漸然全禅繕膳糎噌塑岨措曾曽楚狙	*L:賤/賎
9160	疏疎礎祖租粗素組蘇訴阻遡鼠僧創双叢倉喪壮奏爽宋層匝惣想捜掃挿掻	
9180	操早曹巣槍槽漕燥争痩相窓糟総綜聡草荘葬蒼藻装走送遭鎗霜騒像増憎臓	
91A0	蔵贈造促側則即息捉束測足速俗属賊族続卒袖其揃存孫尊損村遜他多太汰	
91C0	詑唾堕妥惰打柁舵楕陀駄騨体堆対耐岱帯待怠態戴替泰滞胎腿苔袋貸退逮	
91E0	隊黛鯛代台大第醍題鷹滝瀧卓啄宅托択拓沢濯琢託鐸濁諾茸凧蛸只	
9240	叩但達辰奪脱巽竪辿棚谷狸鱈樽誰丹単嘆坦担探旦歎淡湛炭短端箪綻耽胆	
9260	蛋誕鍛団壇弾断暖檀段男談値知地弛恥智池痴稚置致蜘遅馳築畜竹筑蓄	
9280	逐秩窒茶嫡着中仲宙忠抽昼柱注虫衷註酎鋳駐樗瀦猪苧著貯丁兆凋喋寵帖	
92A0	帳庁弔張彫徴懲挑暢朝潮牒町眺聴脹腸蝶調諜超跳銚長頂鳥勅捗直朕沈珍	
92C0	賃鎮陳津墜椎槌追鎚痛通塚栂掴槻佃漬柘辻蔦綴鍔椿潰坪[*M]嬬紬爪吊釣鶴	*M:壺/壷
92E0	亭低停偵剃貞呈堤定帝底庭廷弟悌抵挺提梯汀碇禎程締艇訂諦蹄逓	
9340	邸鄭釘鼎泥摘擢敵滴的笛適鏑溺哲徹撤轍迭鉄典填天展店添纏甜貼転顛点	
9360	伝殿澱田電兎吐堵塗妬屠徒斗杜渡登菟賭途都鍍砥[*N]努度土奴怒倒党冬	*N:礪/砺
9380	凍刀唐塔塘套宕島嶋悼投搭東桃[*O]棟盗淘湯[*P]灯燈当痘祷等答筒糖統到董	*O:檮/梼 *P:濤/涛
93A0	蕩藤討謄豆踏逃透鐙陶頭騰闘働動同堂導憧撞洞瞳童胴萄道銅峠鴇匿得徳	
93C0	涜特督禿篤毒独読栃橡凸突椴届鳶苫寅酉瀞噸屯惇敦沌豚遁頓呑曇鈍奈那	
93E0	内乍凪薙謎灘捺鍋楢馴縄畷南楠軟難汝二尼弐[*Q]匂賑肉虹廿日乳入	*Q:邇/迩
9440	如尿韮任妊忍認濡禰祢寧葱猫熱年念捻撚燃粘乃廼之埜嚢悩濃納能脳膿農	
9460	覗蚤巴把播覇杷波派琶破婆罵芭馬俳廃拝排敗杯盃牌背肺輩配倍培媒梅	
9480	楳煤狽買売賠陪這[*R]秤矧萩伯剥博拍柏泊白箔粕舶薄迫曝漠爆縛莫駁麦函	*R:蠅/蝿
94A0	箱硲箸肇筈櫨幡肌畑畠八鉢溌発醗髪伐罰抜筏閥鳩噺塙蛤隼伴判半反叛帆	
94C0	搬斑板氾汎版犯班畔繁般藩販範釆煩頒飯挽晩番盤磐蕃蛮匪卑否妃庇彼悲	
94E0	扉批披斐比泌疲皮碑秘緋罷肥被誹費避非飛樋簸備尾微枇毘琵眉美	
9540	鼻柊稗匹疋髭彦膝菱肘弼必畢筆逼[*S]姫媛紐百謬俵彪標氷漂瓢票表評豹廟	*S:檜/桧
9560	描病秒苗錨鋲蒜蛭鰭品彬斌浜瀕貧賓頻敏瓶不付埠夫婦富冨布府怖扶敷	
9580	斧普浮父符腐膚芙譜負賦赴阜附侮撫武舞葡蕪部封楓風葺蕗伏副復幅服福	
95A0	腹複覆淵弗払沸仏物鮒分吻噴墳憤扮焚奮粉糞紛雰文聞丙併兵塀幣平弊柄	
95C0	並蔽閉陛米頁僻壁癖碧別瞥蔑箆偏変片篇編辺返遍便勉娩弁鞭保舗鋪圃捕	
95E0	歩甫補輔穂募墓慕戊暮母簿菩倣俸包呆報奉宝峰峯崩庖抱捧放方朋	
9640	法泡烹砲縫胞芳萌蓬蜂褒訪豊邦鋒飽鳳鵬乏亡傍剖坊妨帽忘忙房暴望某棒	
9660	冒紡肪膨謀貌貿鉾防吠頬北僕卜墨撲朴牧睦穆釦勃没殆堀幌奔本翻凡盆	*X:槇/槙
9680	摩磨魔麻埋妹昧枚毎哩[*X]幕膜枕鮪柾鱒桝亦俣又抹末沫迄[*T]繭麿万慢満漫	*T:儘/侭
96A0	蔓味未魅巳箕岬密蜜湊蓑稔脈妙粍民眠務夢無牟矛霧鵡椋婿娘冥名命明盟	
96C0	迷銘鳴姪牝滅免棉綿緬面麺摸模茂妄孟毛猛盲網耗蒙儲木黙目杢勿餅尤戻	
96E0	籾貰問悶紋門匁也冶夜爺耶野弥矢厄役約薬訳躍靖柳[*U]鑓愉愈油癒	*U:藪/薮
9740	諭輸唯佑優勇友宥幽悠憂揖有柚湧涌猶猷由祐裕誘遊邑郵雄融夕予余与誉	
9760	輿預傭幼妖容庸揚揺擁曜楊様洋溶熔用窯羊耀葉蓉要謡踊[*Y]陽養慾抑欲	*Y:遙/遥
9780	沃浴翌翼淀羅螺裸来莱頼雷洛絡落酪乱卵嵐欄濫藍蘭覧利吏履李梨理璃痢	
97A0	裏裡里離陸律率立葎掠略劉流溜琉留硫粒隆竜龍侶慮旅虜了亮僚両凌寮料	
97C0	梁涼猟療瞭稜糧良諒遼量陵領力緑倫厘林淋燐琳臨輪隣鱗麟瑠塁涙累類令	
97E0	伶例冷励嶺怜玲礼苓鈴隷零霊麗齢暦歴列劣烈裂廉恋憐漣煉簾練聯	

シフトJIS	+0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	
9840	蓮	連	錬	呂	魯	櫓	炉	賂	路	露	労	婁	廊	弄	朗	楼	榔	浪	漏	牢	狼	*V	老	聾	蝋	郎	六	麓	禄	肋	録	論	*V:籠/篭
9860	倭	和	話	歪	賄	脇	惑	枠	鷲	亙	亘	鰐	詫	藁	蕨	椀	湾	碗	腕														
9880																																弌	
98A0	丐	丕	个	丱	丶	丼	丿	乂	乖	乘	亂	亅	豫	亊	舒	弍	于	亞	亟	亠	亢	亰	亳	亶	从	仍	仄	仆	仂	仗	仞	仭	
98C0	仟	价	伉	佚	估	佛	佝	佗	佇	佶	侈	侏	侘	佻	佩	佰	侑	佯	來	侖	*t	俔	俟	俎	俘	俛	俑	俚	俐	俤	俥	倚	*t:侭/儘
98E0	倨	倔	倪	倥	倅	伜	俶	倡	倩	倬	俾	俯	們	倆	偃	假	會	偕	偐	偈	做	偖	偬	偸	傀	傚	傅	傴	傲				
9940	僉	僊	傳	僂	僖	僞	僥	僭	僣	僮	價	僵	儉	儁	儂	儖	儕	儔	儚	儡	儺	儷	儼	儻	儿	兀	兒	兌	兔	兢	竸	兩	
9960	兪	兮	冀	冂	囘	册	冉	冏	冑	冓	冕	冖	冤	冦	冢	冩	冪	冫	决	冱	冲	冰	况	冽	凅	凉	凛	几	處	凩	凭		
9980	凰	凵	凾	刄	刋	刔	刎	刧	刪	刮	刳	刹	剏	剄	剋	剌	剞	剔	剪	剴	剩	剳	剿	剽	劍	劔	劒	剱	劈	劑	辨	辧	
99A0	劬	劭	劼	劵	勁	勍	勗	勞	勣	勦	飭	勠	勳	勵	勸	勹	匆	匈	甸	匍	匐	匏	匕	匚	匣	匯	匱	匳	匸	區	卆	卅	
99C0	丗	卉	卍	凖	卞	卩	卮	夘	卻	卷	厂	厖	厠	厦	厥	厮	厰	厶	參	簒	雙	叟	曼	燮	叮	叨	叭	叺	吁	吽	呀	听	
99E0	吭	吼	吮	吶	吩	吝	呎	咏	呵	咎	呟	呱	呷	呰	咒	呻	咀	呶	咄	咐	咆	哇	咢	咸	咥	咬	哄	哈	咨				
9A40	咫	哂	咤	咾	咼	哘	哥	哦	唏	唔	哽	哮	哭	哺	哢	唹	啀	啣	啌	售	啜	啅	啖	啗	唸	唳	啝	喙	喀	咯	喊	喟	
9A60	啻	啾	喘	喞	單	啼	喃	喩	喇	喨	嗚	嗅	嗟	嗄	嗜	嗤	嗔	嘔	嗷	嘖	嗾	嗽	嘛	嗹	噎	噐	營	嘴	嘶	嘲	嘸		
9A80	噫	噤	嘯	噬	噪	嚆	嚀	嚊	嚠	嚔	嚏	嚥	嚮	嚶	嚴	囂	嚼	囁	囃	囀	囈	囎	囑	囓	囗	囮	囹	圀	囿	圄	圉	圈	
9AA0	國	圍	圓	團	圖	嗇	圜	圦	圷	圸	坎	圻	址	坏	坩	埀	垈	坡	坿	垉	垓	垠	垳	垤	垪	垰	埃	埆	埔	埒	埓	堊	
9AC0	埖	埣	堋	堙	堝	塲	堡	塢	塋	塰	毀	塒	堽	塹	墅	墹	墟	墫	墺	壞	墻	墸	墮	壅	壓	壑	壗	壙	壘	壥	壜	壤	
9AE0	壟	壯	*m	壹	壻	壼	壽	夂	夊	夐	夛	梦	夥	夬	夭	夲	夸	夾	竒	奕	奐	奎	奚	奘	奢	奠	奧	奬	奩				*m:壷/壺
9B40	奸	妁	妝	佞	侫	妣	妲	姆	姨	姜	妍	姙	姚	娥	娟	娑	娜	娉	娚	婀	婬	婉	娵	娶	婢	婪	媚	媼	媾	嫋	嫂	媽	
9B60	嫣	嫗	嫦	嫩	嫖	嫺	嫻	嬌	嬋	嬖	嬲	嫐	嬪	嬶	嬾	孃	孅	孀	孑	孕	孚	孛	孥	孩	孰	孳	孵	學	斈	孺	宀		
9B80	它	宦	宸	寃	寇	寉	寔	寐	寤	實	寢	寞	寥	寫	寰	寶	寳	尅	將	專	對	尓	尠	尢	尨	尸	尹	屁	屆	屎	屓	屐	
9BA0	屏	孱	屬	屮	乢	屶	屹	岌	岑	岔	妛	岫	岻	岶	岼	岷	峅	岾	峇	峙	峩	峽	峺	峭	嶌	峪	崋	崕	崗	嵜	崟	崛	
9BC0	崑	崔	崢	崚	崙	崘	嵌	嵒	嵎	嵋	嵬	嵳	嵶	嶇	嶄	嶂	嶢	嶝	嶬	嶮	嶽	嶐	嶷	嶼	巉	巍	巓	巒	巖	巛	巫	已	
9BE0	巵	帋	帚	帙	帑	帛	帶	帷	幄	幃	幀	幎	幗	幔	幟	幢	幤	幇	幵	并	幺	麼	广	庠	廁	廂	廈	廐	廏				
9C40	廖	廣	廝	廚	廛	廢	廡	廨	廩	廬	廱	廳	廰	廴	廸	廾	弃	弉	彝	彜	弋	弑	弖	弩	弭	弸	彁	彈	彌	彎	弯	彑	
9C60	彖	彗	彙	彡	彭	彳	彷	徃	徂	彿	徊	很	徑	徇	從	徙	徘	徠	徨	徭	徼	忖	忻	忤	忸	忱	忝	悳	忿	怡	恠		
9C80	怙	怐	怩	怎	怱	怛	怕	怫	怦	怏	怺	恚	恁	恪	恷	恟	恊	恆	恍	恣	恃	恤	恂	恬	恫	恙	悁	悍	惧	悃	悚	悄	
9CA0	悛	悖	悗	悒	悧	悋	惡	悸	惠	惓	悴	忰	悽	惆	悵	惘	慍	愕	愆	惶	惷	愀	惴	惺	愃	愡	惻	惱	愍	愎	慇	愾	
9CC0	愨	愧	慊	愿	愼	愬	愴	愽	慂	慄	慳	慷	慘	慙	慚	慫	慴	慯	慥	慱	慟	慝	慓	慵	憙	憖	憇	憬	憔	憚	憊	憑	
9CE0	憫	憮	懌	懊	應	懷	懈	懃	懆	憺	懋	罹	懍	懦	懣	懶	懺	懴	懿	懽	懼	懾	戀	戈	戉	戍	戌	戔	戛				
9D40	戞	戡	截	戮	戰	戲	戳	扁	扎	扞	扣	扛	扠	扨	扼	抂	抉	找	抒	抓	抖	拔	抃	抔	拗	拑	抻	拏	拿	拆	擔	拈	
9D60	拜	拌	拊	拂	拇	抛	拉	挌	拮	拱	挧	挂	挈	拯	拵	捐	挾	捍	搜	捏	掖	掎	掀	掫	捶	掣	掏	掉	掟	掵	捫		
9D80	捩	掾	揩	揀	揆	揣	揉	插	揶	揄	搖	搴	搆	搓	搦	搶	攝	搗	搨	搏	摧	摯	摶	摎	*d	撕	撓	撥	撩	撈	撼	據	*d:撹/攪
9DA0	擒	擅	擇	撻	擘	擂	擱	擧	舉	擠	擡	抬	擣	擯	攬	擶	擴	擲	擺	攀	擽	攘	攜	攅	攤	攣	攫	攴	攵	攷	收	攸	
9DC0	畋	效	敖	敕	敍	敘	敞	敝	敲	數	斂	斃	變	斛	斟	斫	斷	旃	旆	旁	旄	旌	旒	旛	旙	无	旡	旱	杲	昊	昃	旻	
9DE0	杳	昵	昶	昴	昜	晏	晄	晉	晁	晞	晝	晤	晧	晨	晟	晢	晰	暃	暈	暎	暉	暄	暘	暝	曁	暹	曉	暾	暼				
9E40	曄	暸	曖	曚	曠	昿	曦	曩	曰	曵	曷	朏	朖	朞	朦	朧	霸	朮	朿	朶	杁	朸	朷	杆	杞	杠	杙	杣	杤	枉	杰	枩	
9E60	杼	杪	枌	枋	枦	枡	枅	枷	柯	枴	柬	枳	柩	枸	柤	柞	柝	柢	柮	枹	柎	柆	柧	*s	栞	框	栩	桀	桍	栲	桎		*s:桧/檜
9E80	梳	栫	桙	档	桷	桿	梟	梏	梭	梔	條	梛	梃	*o	梹	桴	梵	梠	梺	椏	梍	桾	椁	棊	椈	棘	椢	椦	棡	椌	棍	棔	*o:梼/檮
9EA0	棧	棕	椶	椒	椄	棗	棣	椥	棹	棠	棯	椨	椪	椚	椣	椡	棆	楹	楷	楜	楸	楫	楔	楾	楮	椹	楴	椽	楙	椰	楡	楞	
9EC0	楝	榁	楪	榲	榮	槐	榿	槁	槓	榾	槎	寨	槊	槝	榻	槃	榧	樮	榑	榠	榜	榕	榴	槞	槨	樂	樛	槿	權	槹	槲	槧	
9EE0	樅	榱	樞	槭	樔	槫	樊	樒	櫁	樣	樓	橄	樌	橲	樶	橸	橇	橢	橙	橦	橈	樸	樢	檐	檍	檠	檄	檢	檣				
9F40	檗	蘗	檻	櫃	櫂	檸	檳	檬	櫞	櫑	櫟	檪	櫚	櫪	櫻	欅	蘖	櫺	欒	欖	鬱	欟	欸	欷	盜	欹	飮	歇	歃	歉	歐	歙	
9F60	歔	歛	歟	歡	歸	歹	歿	殀	殄	殃	殍	殘	殕	殞	殤	殪	殫	殯	殲	殱	殳	殷	殼	毆	毋	毓	毟	毬	毫	毳	毯		
9F80	麾	氈	氓	气	氛	氤	氣	汞	汕	汢	汪	沂	沍	沚	沁	沛	汾	汨	汳	沒	沐	泄	泱	泓	沽	泗	泅	泝	沮	沱	沾	沺	
9FA0	泛	泯	泙	泪	洟	衍	洶	洫	洽	洸	洙	洵	洳	洒	洌	浣	涓	浤	浚	浹	浙	涎	涕	*p	涅	淹	渕	渊	涵	淇	淦	涸	*p:涛/濤
9FC0	淆	淬	淞	淌	淨	淒	淅	淺	淙	淤	淕	淪	淮	渭	湮	渮	渙	湲	湟	渾	渣	湫	渫	湶	湍	渟	湃	渺	湎	渤	滿	渝	
9FE0	游	溂	溪	溘	滉	溷	滓	溽	溯	滄	溲	滔	滕	溏	溥	滂	溟	潁	漑	*f	滬	滸	滾	漿	滲	漱	滯	漲	滌				*f:潅/灌

シフト JIS	+ 0123456789ABCDEF 1111111111111111 0123456789ABCDEF	
E040	漾漓滷澆潺潸澁澀潯潛濳潭澂潼潘澎澑濂潦澳澣澡澤澹濆澪濟濕濬濔濘濱	
E060	濮濛瀉瀋濺瀑瀁瀏濾瀛瀚潴瀝瀘瀟瀰瀾瀲灑灣炙炒炯烱炬炸炳炮烟烋烝	
E080	烙焉烽焜焙煥煕熈煦煢煌煖煬熏燻熄熕熨熬燗熹熾燒燉燔燎燠燬燧燵燼燹	
E0A0	燿爍爐爛爨爭爬爰爲爻爼爿牀牆牋牘牴牾犂犁犇犒犖犢犧犹犲狃狆狄狎狒	
E0C0	狢狠狡狹狷倏猗猊猜猖猝猴猯猩猥猾獎獏默獗獪獨獰獸獵獻獺珈玳珎玻珀	
E0E0	珥珮珞璢琅瑯琥珸琲琺瑕琿瑟瑙瑁瑜瑩瑰瑣瑪*Z瑾璋璞璧瓊瓏瓔珱	*Z:瑤/瑶
E140	瓠瓣瓧瓩瓮瓲瓰瓱瓸瓷甄甃甅甌甎甍甕甓甞甦甬甼畄畍畊畉畛畆畚畩畤畧	
E160	畫畭畸當疆疇畴疊疉疂疔疚疝疥疣痂疳痃疵疽疸疼疱痍痊痒痙痣痞痾痿	
E180	痼瘁痰痺痲痳瘋瘍瘉瘟瘧瘠瘡瘢瘤瘴瘰瘻癇癈癆癜癘癡癢癨癩癪癧癬癰癲	
E1A0	癶癸發皀皃皈皋皎皖皓皙皚皰皴皸皹皺盂盍盖盒盞盡盥盧盪蘯盻眈眇眄眩	
E1C0	眤眞眥眦眛眷眸睇睚睨睫睛睥睿睾睹瞎瞋瞑瞠瞞瞰瞶瞹瞿瞼瞽瞻矇矍矗矚	
E1E0	矜矣矮矼砌砒*i砠*n硅碎硴碆硼碚碌碣碵碪碯磑磆磋磔碾碼磅磊磬	*i:砿/礦
E240	磧磚磽磴礇礒礑礙礬礫祀祠祗祟祚祕祓祺祿禊禝禧齋禪禮禳禹禺秉秕秧秬	*n:砺/礪
E260	秡秣稈稍稘稙稠稟禀稱稻稾稷穃穗穉穡穢穩龝穰穹穽窈窗窕窘窖窩*e窰	*e:竃/竈
E280	窶竅竄窿邃竇竊竍竏竕竓站竚竝竡竢竦竭竰笂笏笊笆笳笘笙笞笵笨笶筐筺	
E2A0	笄筍笋筌筅筵筥筴筧筰筱筬筮箝箘箟箍箜箚箋箒箏筝箙篋篁篌篏箴篆篝篩	
E2C0	簑簔篦篥*v簀簇簓篳篷簗簍篶簣簧簪簟簷簫簽籌籃籔籏籀籐籘籟籤籖籥籬	*v:篭/籠
E2E0	籵粃粐粤粭粢粫粡粨粳粲粱粮粹粽糀糅糂糘糒糜糢鬻糯糲糴糶糺紆	
E340	紂紜紕紊絅絋紮紲紿紵絆絳絖絎絲絨絮絏絣經綉絛綏絽綛綺綮綣綵緇綽綫	
E360	總綢綯緜綸綟綰緘緝緤緞緻緲緡縅縊縣縡縒縱縟縉縋縢繆繦縻縵縹繃縷	
E380	縲縺繧繝繖繞繙繚繹繪繩繼繻纃緕繽辮繿纈纉續纒纐纓纔纖纎纛纜缸缺罅	
E3A0	罌罍罎罐网罕罔罘罟罠罨罩罧罸羂羆羃羈羇羌羔羞羝羚羣羯羲羹羮羶羸譱	
E3C0	翅翆翊翕翔翡翦翩翳翹飜耆耄耋耒耘耙耜耡耨耿耻聊聆聒聘聚聟聢聨聳聲	
E3E0	聰聶聹聽聿肄肆肅肛肓肚肭冐肬胛胥胙胝胄胚胖脉胯胱脛脩脣脯腋	
E440	隋腆脾腓腑胼腱腮腥腦腴膃膈膊膀膂膠膕膤膣腟膓膩膰膵膾膸膽臀臂膺臉	
E460	臍臑臙臘臈臚臟臠臧臺臻臾舁舂舅與舊舍舐舖舩舫舸舳艀艙艘艝艚艟艤	
E480	艢艨艪艫舮艱艷艸艾芍芒芫芟芻芬苡苣苟苒苴苳苺莓范苻苹苞茆苜茉苙茵	
E4A0	茴茖茲茱荀茹荐荅茯茫茗茘莅莚莪莟莢莖茣莎莇莊荼莵荳荵莠莉莨菴萓菫	
E4C0	菎菽萃菘萋菁菷萇菠菲萍萢萠莽萸蔆菻葭萪萼蕚蒄葷葫蒭葮蒂葩葆萬葯葹	
E4E0	萵蓊葢蒹蒿蒟蓙蓍蒻蓚蓐蓁蓆蓖蒡蔡蓿蓴蔗蔘蔬蔟蔕蔔蓼蕀蕣蕘蕈	*j:蕊/蘂
E540	蕁*j蕋蕕薀薤薈薑薊薨蕭薔薛*u薇薜蕷蕾薐藉薺藏薹藐藕藝藥藜藹蘊蘓蘋	*u:薮/藪
E560	藾藺蘆蘢蘚蘰蘿虍乕虔號虧虱蚓蚣蚩蚪蚋蚌蚶蚯蛄蛆蚰蛉*c蚫蛔蛞蛩蛬	*c:蛎/蠣
E580	蛟蛛蛯蜒蜆蜈蜀蜃蛻蜑蜉蜍蛹蜊蜴蜿蜷蜻蜥蜩蜚蝠蝟蝸蝌蝎蝴蝗蝨蝮蝙蝓	
E5A0	蝣蝪*r螢螟螂螯蟋螽蟀蟐雖螫蟄螳蟇蟆螻蟯蟲蟠蠏蠍蟾蟶蟷蠎蟒蠑蠖蠕蠢	*r:蝿/蠅
E5C0	蠡蠱蠶蠹蠧蠻衄衂衒衙衞衢衫袁衾袞衵衽袵衲袂袗袒袮袙袢袍袤袰袿袱裃	
E5E0	裄裔裘裙裝裹褂裼裴裨裲褄褌褊褓襃褞褥褪褫襁襄褻褶褸襌褝襠襞	
E640	襦襤襭襪襯襴襷襾覃覈覊覓覘覡覩覦覬覯覲覺覽覿觀觚觜觝觧觴觸訃訖訐	
E660	訌訛訝訥訶詁詛詒詆詈詼詭詬詢誅誂誄誨誡誑誥誦誚誣諄諍諂諚*g諳諧	*g:諌/諫
E680	諤諱謔諠諢諷諞諛謌謇謚諡謖謐謗謠謳鞫謦謫謾謨譁譌譏譎證譖譛譚譫譟	
E6A0	譬譯譴譽讀讌讎讒讓讖讙讚谺豁谿豈豌豎豐豕豢豬豸豺貂貉貅貊貍貎貔豼	
E6C0	貘戝貭貪貽貲貳貮貶賈賁*l賣賚賽賺賻贄贅贊贇贏贍贐齎贓賍贔贖赧赭赱	*l:賎/賤
E6E0	赳趁趙跂趾趺跏跚跖跌跛跋跪跫跟跣跼踈踉跿踝踞踐踟蹂踵踰踴蹊	
E740	蹇蹉蹌蹐蹈蹙蹤蹠踪蹣蹕蹶蹲蹼躁躇躅躄躋躊躓躑躔躙躪躡躬躰軆躱躾軅	
E760	軈軋軛軣軼軻軫軾輊輅輕輒輙輓輜輟輛輌輦輳輻輹轅轂輾轌轉轆轎轗轜	
E780	轢轣轤辜辟辣辭辯辷迚迥迢迪迯*q迴逅迹迺逑逕逡逍逞逖逋逧逶逵逹迸遏	*q:迩/邇
E7A0	遐遑遒逎遉逾遖遘遞遨遯遶隨遲邂遽邁邀邊邉邏邨邯邱邵郢郤扈郛鄂鄒鄙	
E7C0	鄲鄰酊酖酘酣酥酩酳酲醋醉醂醢醫醯醪醵醴醺釀釁釉釋釐釖釟釡釛釼釵釶	
E7E0	鈞釿鈔鈬鈕鈑鉞鉗鉅鉉鉤鉈銕鈿鉋鉐銜銖銓銛鉚鋏銹銷鋩錏鋺鍄錮	

シフト	+																1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
JIS	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
E840	錙	錢	錚	錣	錺	錵	錻	鍜	鍠	鍼	鍮	鍖	鎰	鎬	鎭	鎔	鎹	鏖	鏗	鏨	鏥	鏘	鏃	鏝	鏐	鏈	鏤	鐚	鐔	鐓	鐃	鐇
E860	鐐	鐶	鐫	鐵	鐡	鐺	鑁	鑒	鑄	鑛	鑠	鑢	鑞	鑪	鈩	鑰	鑵	鑷	鑽	鑚	鑼	鑾	钁	鑿	閂	閇	閊	閔	閖	閘	閙	
E880	閠	閨	閧	閭	閼	閻	閹	閾	闊	濶	闃	闍	闌	闕	闔	闖	關	闡	闥	闢	阡	阨	阮	阯	陂	陌	陏	陋	陷	陜	陞	陝
E8A0	陟	陦	陲	陬	隍	隘	隕	隗	險	隧	隱	隲	隰	隴	隶	隸	隹	雎	雋	雉	雍	襍	雜	霍	雕	雹	霄	霆	霈	霓	霎	霑
E8C0	霏	霖	霙	霤	霪	霰	霹	霽	霾	靄	靆	靈	靂	靉	靜	靠	靤	靦	靨	勒	靫	*k	靹	鞅	靼	鞁	靺	鞆	鞋	鞏	鞐	鞜
E8E0	鞨	鞦	鞣	鞳	鞴	韃	韆	韈	韋	韜	韭	齏	韲	竟	韶	韵	頏	頌	*h	頤	頡	頷	頽	顆	顏	顋	顫	顯	顰			
E940	顱	顴	顳	颪	颯	颱	颶	飄	飃	飆	飩	飫	餃	餉	餒	餔	餘	餡	餝	餞	餤	餠	餬	餮	餽	餾	饂	饉	饅	饐	饋	饑
E960	饒	饌	饕	馗	馘	馥	馭	馮	馼	駟	駛	駝	駘	駑	駭	駮	駱	駲	駻	駸	騁	騏	騅	駢	騙	騫	騷	驅	驂	驀	驃	
E980	騾	驕	驍	驛	驗	驟	驢	驥	驤	驩	驫	驪	骭	骰	骼	髀	髏	髑	髓	體	髞	髟	髢	髣	髦	髯	髫	髮	髴	髱	髷	髻
E9A0	鬆	鬘	鬚	鬟	鬢	鬣	鬥	鬧	鬨	鬩	鬪	鬮	鬯	鬲	魄	魃	魏	魍	魎	魑	魘	魴	鮓	鮃	鮑	鮖	鮗	鮟	鮠	鮨	鮴	鯀
E9C0	鯊	鮹	鯆	鯏	鯑	鯒	鯣	鯢	鯤	鯔	鯡	*a	鯲	鯱	鯰	鰕	鰔	鰉	鰓	鰌	鰆	鰈	鰒	鰊	鰄	鰮	鰛	鰥	鰤	鰡	鰰	鱇
E9E0	鰲	鱆	鰾	鱚	鱠	鱧	鱶	鱸	鳧	鳬	鳰	鴉	鴈	鳫	鴃	鴆	鴪	鴦	*b	鴣	鴟	鵄	鴕	鴒	鵁	鴿	鴾	鵆	鵈			
EA40	鵝	鵞	鵤	鵑	鵐	鵙	鵲	鶉	鶇	鶫	鵯	鵺	鶚	鶤	鶩	鶲	鷄	鷁	鶻	鶸	鶺	鷆	鷏	鷂	鷙	鷓	鷸	鷦	鷭	鷯	鷽	鸚
EA60	鸛	鸞	鹵	鹹	鹽	麁	麈	麋	麌	麒	麕	麑	麝	麥	麩	麸	麪	麭	靡	黌	黎	黏	黐	黔	黜	點	黝	黠	黥	黨	黯	
EA80	黴	黶	黷	黹	黻	黼	黽	鼇	鼈	皷	鼕	鼡	鼬	鼾	齊	齒	齔	齣	齟	齠	齡	齦	齧	齬	齪	齷	齲	齶	龕	龜	龠	*w
EAA0	*x	*y	*z	*3	*3																											
EAC0																																
EAE0																																
EB40																																
:																																
EFE0																																
F040	*4 ユーザー外字域 ...																															
F060	:																															
F080	:																															
:																																
F9E0																																
FA40																					*1							*2				
FA60																																
FA80																																
FAA0																																
FAC0																																
FAE0																																
FB40																																
FB60																																
FB80																																
FBA0																																
FBC0																																
FBE0																																
FC40																																
FC60																																
FC80																																
FCA0																																
FCC0																																
FCE0																																

*k:靭/靱
*h:頚/頸
*a:鯵/鰺
*b:鴬/鶯
*w:尭/堯
*x:槙/槇
*y:遥/遙
*z:瑶/瑤
*1:¬
*2:∵

テクニカル・リファレンス

DOS/V オプション機能

技術解説編

特記事項

「DOS/V オプション機能 技術解説編」は、OADGがオプションとして定義しているDOS/Vの拡張機能を記述しています。これらの拡張機能の中には、DOS/VのV-Text環境で正しく動作するためのアプリケーション・プログラム・インターフェースやビデオ拡張ドライバーあるいはプリンター・ドライバーを記述するためのドライバー・インターフェースなどが含まれています。これらの拡張機能を使用することによって、より機能性の高いアプリケーション・プログラムを作成したり、特定ハードウェアをサポートするドライバーを開発することが可能となります。ただし、これらの拡張機能は、対象とするOADG DOS/Vおよびそのバージョンによっては必ずしも提供されない場合があります。アプリケーション・プログラムあるいはドライバーの作成にあたっては、このことを考慮したうえでオプション機能をご使用になることをお勧めします。

本書は、インターナショナル・ビジネス・マシーンズ・コーポレーション(以下IBMと略します)の著作物です。これはIBM社の以下の出版物の内容の一部を記載しており、これらの著作権はIBMが所有しています。

- PC DOS J6.1/V BIOS インターフェース技術解説書
- IBM DOS/V Extension V2.0 ユーザーズ・ガイド

これらの内容の無断転載、複製などの著作権に抵触する使用はできません。

目次

第1章　フォントサブシステムの拡張機能

この章では、DOS/Vのフォント・サブシステムに対してオプションとして定義されている付加機能と、それらを使用する上での考慮点について説明します。これらの拡張された機能には、DOS/V上でV-Textの特長の一つであるフォント書体の切り換えなどを実現するための機能などが含まれています。

フォント・サブシステムの拡張API

フォント・サブシステムに対して付加されている拡張アプリケーション・プログラム・インターフェース(API)は以下のとおりです。

INT 15H フォント入出力

AX=500xH フォントの読み取りと書き込み

文字パターンの‘読み取り’と‘書き込み’を行うために、‘フォントの読み取り’と‘フォントの書き込み’機能のアドレスを取得します。その際、アドレスを取得するフォントのフォント・インデックスを指定することができます。フォント・インデックスについては、5-2ページの『[Font]セクション』を参照してください。

各アドレスを読み取るために、次の設定が必要です。

‘フォントの読み取り’機能のアドレスの取得

＜設定＞

AL = 00H

(BH)　フォントの種類

　ビット0 = 0　1バイト文字セット

　　　　= 1　2バイト文字セット

　ビット1~7　予約済み

(BL)　フォント・インデックス

(DH, DL) フォント・サイズ（文字幅、文字の高さ）

(BP)　コード・ページ（『コード・ページ』を参照）

＜結果＞

CF　キャリー・フラグ

　= 0　正常に完了（AH=0の場合）

　= 1　エラーにより未完了

(AH)　完了コード（『完了コード』を参照）

(ES:BX) 指定した機能のアドレス（AH=0の場合）

‘フォントの書き込み’機能のアドレスの取得

＜設定＞

AL = 01H

(BH)　フォントの種類
　ビット0 = 0　1バイト文字セット
　= 1　2バイト文字セット
　ビット1~7　予約済み
(BL)　フォント・インデックス
(DH, DL) フォント・サイズ（文字の幅、文字の高さ）
(BP)　コード・ページ（『コード・ページ』を参照）

＜結果＞
CF　キャリー・フラグ
　= 0　正常に完了（AH=0の場合）
　= 1　エラーにより未完了
(AH)　完了コード（『完了コード』を参照）
(ES:BX) 指定した機能のアドレス（AH=0の場合）

コード・ページ:

指定できるコード・ページは、以下のとおりです。

0　コード・ページの省略値

完了コード:

返されるコードは、以下のとおりです。

00H　正常に完了
01H　無効なフォントの種類が指定された（BHの指定が無効）
02H　BLの指定が無効です
03H　無効なフォント・サイズが指定された（DXの指定が無効）
04H　無効なコード・ページが指定された（BPの指定が無効）
86H　機能はサポートされていない

文字イメージの読み出しと書き込み

‘フォントの読み取り’機能と‘フォントの書き込み’機能の具体的な呼び出し方法については、「DOS/V BIOS インターフェース 技術解説編」を参照してください。

第2章 ビデオ・サブシステムの拡張機能

この章では、DOS/Vのビデオ・サブシステム(共通サブシステムおよびビデオ拡張ドライバー)に対してオプションとして定義されている付加機能と、それらを使用する上での考慮点について説明します。これらの拡張された機能には、DOS/V上で高品位テキスト・モードやワイド・テキスト・モード、あるいは縦書きモードといったV-Text環境を実現するための諸機能が含まれています。

ビデオ・サブシステムの拡張API

ビデオ・サブシステムに対して付加された拡張アプリケーション・プログラム・インターフェース(API)は以下のとおりです。

INT 10H ディスプレイ入出力

AH=00H モード設定

<設定>

(AL)= 70H　可変サイズのCGAテキスト・モード

注:

1. テキスト・バッファーの文字と属性フォーマットはビデオ・モード03Hと同じですが、桁数と行数はビデオ・コントローラー、使用しているディスプレイ、またはビデオBIOS (またはドライバー) よって変わります。現在、使用可能なテキスト画面サイズは、「DOS/Vモード設定情報の取得」機能 (INT10H, AH=12H, BL=3AH) と「DOS/V拡張モード・テーブルの取得」機能(INT10H, AH=11H, BL=31H)によって参照することができます。
2. 特定の拡張テキスト・モードが、「DOS/Vテキスト密度の切り替え」機能 (INT10H, AH=12H, BL=39H) によって、現在モード70Hに対して設定されている場合のみ有効です。拡張テキスト・モードが設定されていないとモードは変更されません。

(AL) = 71H　可変サイズの拡張CGAテキスト・モード

注:

1. テキスト・バッファーの文字と属性フォーマットはビデオ・モード73Hと同じですが、桁数と行数はビデオ・コントローラー、使用しているディスプレイ、またはビデオBIOS (またはドライバー) よって変わります。現在、使用可能なテキスト画面サイズは、「DOS/Vモード設定情報の取得」機能 (INT10H, AH=12H, BL=3AH) と「DOS/V拡張モード・テーブルの取得」機能(INT10H, AH=11H, AL=31H)によって参照することができます。

2. 特定の拡張テキスト・モードが、「DOS/Vテキスト密度の切り替え」機能(INT10H, AH=12H, BL=39H) によって、現在モード71Hに対して設定されている場合のみ有効です。拡張テキスト・モードが設定されていないとモードは変更されません。

AH = 11H 文字ジェネレーター

(AL) = 18H　DOS/V現行高密度テキスト・フォントのロード（モード03Hと73Hの場合のみ）:

<設定>

(BL) = 設定するブロック（常に0）

注:

1. この機能は、標準のDOS/Vテキスト・モード（80x25) から現行の高密度テキスト・モード（80x26以上）へ変えるために使用します。行数は変わりますが、桁数は変わりません。現行の高密度テキスト・モードは、「DOS/Vテキスト密度の切り替え」機能(INT10H, AH=12H, BL=39H)によって設定されます。
2. この機能は、「モード設定」機能(INT10H, AH=00H)の直後に呼び出されなければなりません。そうでない場合は、結果は保証されません。
3. 特定の拡張テキスト・モードが、「DOS/Vテキスト密度の切り替え」機能(INT10H, AH=12, BL=39H)によって、現行モードに対して設定されている場合のみ有効です。拡張テキスト・モードが設定されていないと高密度テキスト・モードに変更されません。

(AL) = 31H　DOS/V拡張モード・テーブルの取得:

<結果>

(ES:BP) = 拡張モード・テーブルのアドレス

(CX) = エントリー数

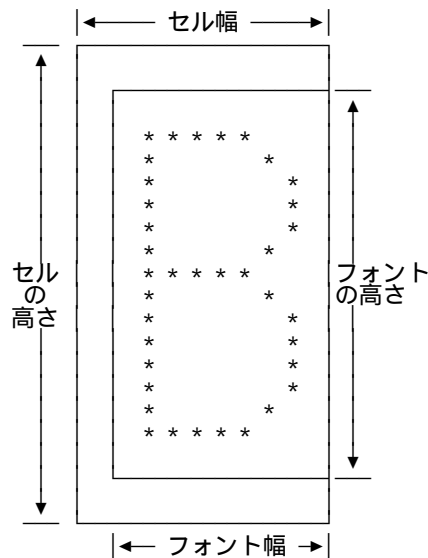
テーブルのそれぞれのエントリーは16バイトで、次のようなフォーマットで並んでいます。

バイト00H　モード番号
バイト01H　モード情報
バイト02H　桁数
バイト03H　行数
バイト04H　文字セルの幅
バイト05H　文字セルの高さ
バイト06H　文字フォントの幅
バイト07H　文字フォントの高さ
ワード08H　予約済み(システムで使用されます)
ワード0AH　予約済み(システムで使用されます)
バイト0CH　予約済み（システムで使用されます）

バイト0DH　予約済み（システムで使用されます）
バイト0EH　予約済み（システムで使用されます）
バイト0FH　予約済み

注:

1. このテーブルはDOS/V拡張テキスト・モード（高品位テキスト・モードと可変密度テキスト・モード）用のものです。
2. モード番号が03Hまたは73Hの場合は、桁数は常に80ですが、行数は26以上です。
3. モード番号が70Hまたは71Hの場合は、桁数も行数も可変です。
4. モード情報は次のように定義されています。

5. セルの幅と高さ、フォントの幅と高さは、次のようなマトリックスで示されます。フォント外のスペースは縦または横の罫線、下線、および文字と文字の間をあけるためのものです。

6. ビデオ拡張ドライバーが現在登録されていなければ、レジスターの内容は変更されません。

AH = 12H 代替選択

(BL) = 38H　DOS/Vフォント・サイズの切り替え (モード03Hと73Hの場合のみ):

<設定>

(AL) = 拡張モード・テーブルのインデックス(0,1,...)
または-1 (FFH) (基本(従来)テキスト・モード(80x25)に戻す場合)

(BH) = モード番号((AL)= -1の場合)

<結果>

(AL) = 12H　　この機能がサポートされている
上記以外　この機能がサポートされていない

注:

1. この機能により設定されたモードは、次回の「モード設定」機能(INT10H, AH=00H)で有効となります（モード03Hと73Hのみ）。この設定は、異なったパラメーターで、再度この機能が呼び出されるまで有効となります。
2. 拡張モード・テーブルのインデックスは、「DOS/V拡張モード・テーブルの取得」機能 (INT10H, AH=11H, AL=31H)によって取得したテーブルのインデックスです。(AL)で指定されるモードは、80x25のモード03H、または80x25のモード73Hのどちらかでなければなりません。
3. 拡張モード・テーブルのインデックスが-1 (FFH) ならば、(BH)で指定されたモードのフォント・サイズの設定は標準設定に戻ります。
4. フォント・サイズの切り替えは、モード03Hと73Hの間とで別々に設定できます。

(BL) = 39H　DOS/Vテキスト密度の切り替え:

<設定>

(AL) = 拡張モード・テーブルのインデックス (0,1,...)
または-1 (FFH) (この設定を解除する場合)

(BH) = モード番号((AL) = -1の場合)

<結果>

(AL) = 12H　　この機能がサポートされている
上記以外　この機能がサポートされていない

注:

1. この機能により設定されたモードは、モード70Hと71Hの場合が「モード設定」機能 (INT10H, AH=00H)で、モード03Hと73Hの場合が「現行高密度テキスト・フォントのロード」機能(INT10H, AH=11H, AL=18H)で有効となります。この設定は、異なったパラメーターで、再度この機能が呼び出されるまで有効です。

2. 拡張モード・テーブルのインデックスは、「DOS/V拡張モード・テーブルの取得」機能 (INT10H, AH=11H, AL=31H)によって取得したテーブルのインデックスです。

3. 拡張モード・テーブル・インデックスが -1 (FFH)の場合は、(BH)で指定されたモードのテキスト密度の設定は解除されます。

4. テキスト密度の設定は03H、73H、70H、71Hの間で別々に設定できます。

(BL) = 3AH　DOS/Vモード設定情報の取得:

<設定>

(AL) = モード番号 (03H, 73H, 70H, または 71H)

<結果>

(AL) = 12H　　この機能がサポートされている
上記以外　この機能がサポートされていない

(CH) = フォント・サイズの設定(モード 03H と 73H のみ)
拡張モード・テーブルのインデックス (0,1....) または -1 (FFH)

(CL) = テキスト密度の設定
拡張モード・テーブルのインデックス (0,1,...) または -1 (FFH)

注:

1. フォント・サイズの設定は、入力の(AL)の値が03Hか73Hのどちらかである場合のみ有効です。-1 (FFH) はモード03Hまたは73Hに対して標準フォント・サイズ (8 x 19)のテキスト・モードが設定されていることを表しています。

2. -1 (FFH) がテキスト密度の設定として戻される場合は、入力の (AL)で指定されたモードには、可変密度テキスト・モードは設定されていないことを表しています。

AH=1DH　キーボード・シフト状況表示域の制御

(AL) = 03H キーボード・シフト状況表示方法の選択

＜設定＞

(BX)

=0	25行モード
=1	26行モード
=他の値	予約済み

(AL) = 04H キーボード・シフト状況表示方法の取得

＜結果＞

(BX)

=0　25行モード

=1　26行モード

注:

1. この機能は日本語入力支援システム（$IAS.SYS）に予約済みです。アプリケーション・プログラムは、この機能を呼びだしてキーボード・シフト標識域の表示や消去を行ってはなりません。シフト標識域の表示／非表示をアプリケーション・プログラムが行う場合はINT16h（AH=14h）を使用してください。

2. 画面下部が標識域行に予約されると、‘画面の行数−1（BIOSデータ域0040:0084（バイト））’が(BX)に指定された行数分減ります。

3. "AH=00H ビデオ・モードの設定"において、キーボード・シフト標識域は、自動的に消去されますが、日本語入力支援システムが導入されている場合は、この機能により、ビデオ・モードの設定前と設定後もシフト標識域はそのまま維持されます。

4. "AL=03H キーボード・シフト状況表示方法の選択"および"AL=04H キーボード・シフト状況表示方法の取得"は、ビデオ・モード03Hおよび73Hに対してのみ意味を持ちます。

5. 実際に26行モードに変わったかどうかは、以下の方法で確認してください。

```
mov   ax, 1D03h       ; 26行モードにスイッチ
mov   bx, 1           ;
int   10h             ;
mov   ax, 1D04h       ; テキスト行数の読み取り
mov   bx, 0           ; あらかじめ0にセット
int   10h             ;
cmp   bx, 1           ; リクエストが成功したかどうか確認
jne   fixed_to_25     ; 失敗であればテキスト行数は25
```

AH=FEH ビデオ・バッファー・アドレスの読み取り

＜設定＞

(ES:DI) ビデオ・バッファーのアドレスを指定します。
= B800:0000H

＜結果＞

(ES:DI) ビデオ・バッファーの実際のアドレス

ハードウェア・ビデオ・バッファーが存在する場合、このアドレスは不変です。存在しない場合、擬似ビデオ・バッファーのアドレスに変ります。

注: この機能はビデオ・モード03H, 70H, 71H, および73Hで使用可能です。ただし、古いバージョンのBIOSでは、この機能はモード03Hでのみ使用可能です。モード70H, 71H, 73Hで使用可能かどうか調べる必要がある場合は、下記のプログラムに従ってください。

```
        ; 目的のモードに設定
        mov     ax, 00xxh
        int     10h
        ; モード変更は成功したか
        mov     ah, 0Fh
        int     10h
        cmp     al, xxh
        jne     mode_not_supported
        ; ビデオ・バッファー・アドレスの取得
        mov     di, B800h
        mov     es, di
        xor     di, di
        mov     ah,0FEh
        int     10h
        ; 取得したビデオ・バッファー・アドレスは有効か？
        cmp     word ptr es:ffdi[,0720h
        jne     address_invalid
        mov     byte ptr es:ffdi + 01h[,00h
        mov     ah, 08h
        xor     bh, bh
        int     10h
        cmp     ax, 0020h
        mov     byte ptr ffdi + 01[,07h
        jne     address_invalid
        ;
        ; 取得したビデオ・バッファー・アドレスは有効か？
        ;
address_invalid:
        ;
mode_not_supported:
```

AH=FFH 画面表示の更新

＜設定＞

(ES:DI) 更新されるビデオ・バッファー内の先頭アドレスを設定します。

(CX) ビデオ・バッファーに書き込まれる文字数を指定します。

注:

1. この機能はビデオ・モード03H，70H，71H，および73Hで使用可能です。ただし、古いバージョンのBIOSでは、この機能はモード03Hでのみ使用可能です。モード70H，71H，73Hで使用可能かどうか調べる必要がある場合は、INT 10H, AH=FEHの **注:** のプログラムに従ってください。

2. ハードウェア・ビデオ・バッファーが存在する場合、この機能は効果がありません。

アプリケーション・プログラミング・ガイドライン

モード番号と画面サイズの取得

次の内容は、DOS/V BIOSインターフェースにあまり詳しくない方のための情報です。同じ機能を実行するための他の方法もありますが、次のコードを１つのサンプルとして示します。ここで取得した情報は、以降のサンプル・コード中でも使用します。

```
get_mode_info   proc    near
                push    es
                push    ax
                push    bx
                push    cx
                push    dx
                push    bp
                mov     ah, 0FH         ; ディスプレイ状況の取得
                int     10H             ;
                and     al, not 80H     ; モード = (AL)の(B6-B0)
                mov     crt_mode, al    ;
                mov     crt_cols, ah    ; 桁数を保管
                mov     ax, 1130H       ; SBCS フォント情報の取得
                mov     bh, 01H         ;
                int     10H             ; 行数 - 1 -> DL
                inc     dl              ; 行数 -> DL
                mov     crt_rows, dl    ; スクロールする範囲の行数を保管
                mov     ax, 1D02H       ; キーボード・シフト標識域の行数
                                        ; の取得
                xor     bx, bx          ; 0に設定
                int     10H             ; キーボード・シフト標識域の行数
                                        ; の取得 -> BX
                add     dl, bl          ; 画面全体の行数 -> DL
                mov     total_rows, dl  ; 画面全体の行数を保管
                pop     bp
                pop     dx
                pop     cx
                pop     bx
                pop     ax
                pop     es
get_mode_info   endp
```

アプリケーション・プログラムの類別

DOS/Vビデオ拡張機能では、高品位テキスト・モードと可変密度テキスト・モードの２つが提供されます（同時に両方のモードを提供することもあります。この場合には、以下の説明で可変密度テキスト・モードとして扱ってください)。高品位テキスト・モードは、アプリケーションに対して透過です(アプリケーションの変更、または特別なサポートは必要ありません)。可変密度テキスト・モードは、アプリケーションによる対応が必要です。現行の、およびこれからのDOS/Vテキスト・アプリケーションは、次の４つに類別することができます。

1. TTYタイプのアプリケーション

2. 可変密度テキスト・モードをサポートしていないアプリケーション

3. 可変密度テキスト・モードをサポートしているが、専用のAPIを使用しないアプリケーション

4. 可変密度テキスト・モードをサポートし、専用のAPIを使用するアプリケーション

1. TTYタイプのアプリケーション

TTYタイプのアプリケーションとは、ここでは、画面に文字を表示する場合に、文字コードのみをBIOSに指定し、表示位置はBIOSに任せるようなアプリケーションを意味します。

この種類のアプリケーションは、DOS/V Video Extensionの観点からすると、最も「作法の良い」アプリケーションで、特別なガイド・ラインはありません。ただし、桁数を80に固定して仮定しないでください。現行の位置を画面上で次の行に動かすには、最終桁までスペース文字を埋める代わりに、復帰と改行を使用してください。

2. 可変密度テキスト・モードをサポートしていないアプリケーション

この種類のアプリケーションは、起動時に通常03Hか73Hのモードのどちらかに設定しますので、常に80x25のテキスト・モードで期待どうりに動作します。これは、テキスト密度の設定にかかわらず、03Hと73Hのモードに設定することによって、画面サイズが必ず80x25になるからです。「作法の良い」アプリケーションは終了する場合は、起動前のモードに戻します。

3. 可変密度テキスト・モードをサポートしているが、専用のAPIを使用しないアプリケーション

この種類のアプリケーションは、現行のモードがグラフィックス・モードでない限り、モードを設定しません。現行の画面サイズを調べ、その画面サイズに従ってテキストまたはパネルを画面に表示します。終了する際は、モードを設定する代わりに画面を消去します。この種類のアプリケーションで典型的なものは、テキスト・エディターとワード・プロセッサーです。

次のサンプル・コードは、このようなアプリケーションの開始と終了の典型的な例を示しています。

(開始)

```
        call    get_mode_info   ; モード番号と画面サイズの取得
        call    clear_screen
```

(終了)

```
        call    clear_screen


clear_screen    proc    near
        push    ax
        push    bx
        push    cx
        push    dx
        mov     ax, 0600H       ; 画面全体を上にスクロール
        mov     bh, 07H         ; 前景＝白／背景＝黒
        xor     cx, cx          ; (0,0)-------------+
        mov     dh, crt_rows    ;   |               |
        mov     dl, crt_cols    ;   |               |
        sub     dx, 0101H       ;   +----(crt_rows-1, crt_cols-1)
        int     10H             ; 画面を消去
        pop     dx
        pop     cx
        pop     bx
        pop     ax
        ret
clear_screen    endp
```

4. 可変密度テキスト・モードをサポートし、専用のAPIを使用するアプリケーション

この種類のアプリケーションは、起動時にモードを設定します。このアプリケーションがモード03Hまたは73Hを使用している場合は、高密度テキスト・モードに変えるために「DOS/V現行高密度テキスト・フォントのロード」機能を呼び出す必要があります。モード70Hまたは71Hの場合は、そのモード自体が高(可変)密度テキスト・モードなので、モード設定後に別の機能を呼び出す必要はありません。

次のサンプル・コードは、モード03Hと73Hの高密度テキスト・アプリケーションの典型的な開始の例を示しています。

(開始)

```
    call    get_mode_info   ; モード番号と画面サイズの取得
    mov     ax, 0003H       ; CGA テキスト・モード
                            ; 0073H (拡張CGAテキスト・モードの場合)
    int     10H             ; モードの設定
    mov     ax, 1118H       ; 現行の高密度テキスト・フォントのロード
    xor     bl, bl          ; ブロック=0
    int     10H             ; 高密度テキスト・モードへの変更
```

次のコードは、モード70Hと71Hの可変密度テキスト・アプリケーションの典型的な開始の例を示しています。

(開始)

```
        call    get_mode_info    ; モード番号と画面サイズの取得
        mov     ax, 0070H        ; 可変 CGA テキスト・モード
                                 ; 0071H(可変拡張CGAテキスト・
                                 ; モードの場合)
        int     10H              ; モードの設定
```

これらの可変密度テキスト・モードのどれかに設定する前に、そのモードが有効かどうかを、次のようにチェックすることができます。

```
        mov     ah, 12H          ;
        mov     bl, 3AH          ; DOS/Vモード設定情報の取得
        mov     al, 03H          ; モード: 03H, 73H, 70H, 71Hのいずれか
        int     10h              ;
        cmp     al, 12h          ; この機能がサポートされているか?
;       $if     e,  and          ;
        jne     $$IF0            ;
        cmp     cl, 0FFH         ; 有効か?
;       $if     ne               ; この機能がサポートされていて、かつ有効な
        je      $$IF0            ; 場合は処理を続行
                ; available      ;
;       $else                    ; そうでない場合はエラー処理
        jmp     $$EN0            ;
$$IF0:                           ;
                ; not available ;
;       $endif                   ;
$$EN0:                           ;
```

ビデオ拡張ドライバーが現在登録されているのかどうか、または「DOS/V拡張モード・テーブルの取得」機能 (INT10H, AX=1131H)によって取得したモード・テーブルが有効かどうかを知る必要がある場合は、次のようにしてチェックすることができます。

```
        mov     ax, 1131H        ; DOS/V拡張モード・テーブルの取得
        xor     cx, cx           ; 0に設定
        int     10H              ;
        or      cx, cx           ; 1つ以上の拡張モードのエントリーがあるか?
;       $if     nz               ;
        je      $$IF1            ;
                ; 1つ以上の拡張モードのエントリー、または
                ; ビデオ拡張ドライバーがある。
;       $else                    ;
        jmp     $$EN1            ;
$$IF1:
                ; 拡張モードのエントリー、および
                ; ビデオ拡張ドライバーがない。
;       $endif
$$EN1:
```

アプリケーション・プログラムが終了する場合は、画面モードとテキストの密度を元に戻すようにしてください。次のサンプル・コードは、画面モードとテキストの密度を元に戻す例を示しています。

(終了)

```
      ; 画面モードを元に戻す
      xor     ah, ah          ; モードを設定
      mov     al, crt_mode    ; 元の画面モード
      int     10H             ;
      ;  元の画面モードが03Hと73Hの場合、テキスト密度を元に戻す
      cmp     total_rows, 25  ;
;     $if     ne, and         ; 可変密度モードの場合、および
      je      $$IF2           ;
      mov     al, crt_mode    ;
      cmp     al, 03H         ;
;     $if     e, or           ; 元のモード=CGAテキスト・モードの場合、
      je      $$LL2           ;
      cmp     al, 73H         ; または
;     $if     e               ; 元のモード=拡張CGAテキスト・モードの場合
      jne     $$EN2           ;
$$LL2:                        ;
          mov     ax, 1118H   ;
          xor     bl, bl      ;
          int     10H         ; 高密度テキスト・フォントをロード
;     $endif                  ;
$$IF2:                        ;
$$EN2:                        ;
```

コマンド・シェル・プロセスを起動するアプリケーション

これまでに説明してきたカテゴリー1から3のアプリケーション・プログラムは、基本的に特定の画面モードおよび特定のテキスト密度に依存しないので、ここでは主にカテゴリー４のアプリケーション・プログラムの説明をします。

アプリケーション・プログラムが、可変密度テキスト・モードで実行中にコマンド・シェル・プロセスを起動した場合(たとえば、エディターの中からコマンド・プロンプトに入った場合)、ユーザーはテキストの密度の観点から次の操作のいずれかをすることができます。

- 何もしない
- 画面モードを変更
- テキスト密度の設定を変更
- ビデオ拡張ドライバーの除去

上記の「何もしない」以外の操作によって、現行のテキスト密度、または現行の画面モードが変更される可能性があります。したがって、制御がアプリケーション・プログラムに戻るときに、アプリケーション・プログラムはその変更に対処する必要があります。この場合は、次のどちらかの方法で対処してください。

- 現行のテキスト密度を受け入れ、そのまま処理を続行する。

- 以前のテキスト密度が利用可能かどうかを検査し、可能な場合は、そのモードに設定する。可能でない場合は、動作可能な適当な画面モード(標準テキスト・モード03Hまたは73Hなど)に変更する。

ビデオ拡張ドライバー・インターフェース

この章では、ビデオ拡張ドライバーまたはシステム・プログラムを開発する方を対象に、ビデオ拡張ドライバーとのインターフェースについて説明しています。アプリケーション・プログラムを開発する方は、ここで述べるドライバー・インターフェースではなく、『ビデオ・サブシステムの拡張API』で説明したBIOS INT10H機能をお使いください。

拡張されたビデオ・サブシステムの構成

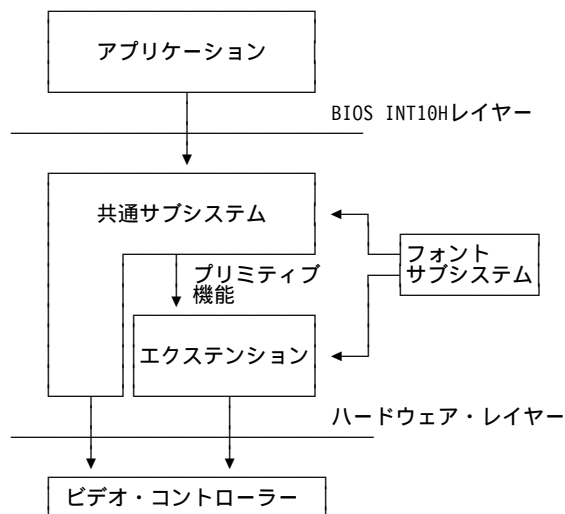



図　2-1. 拡張されたビデオ・サブシステムの構成

DOS/V 拡張ビデオ・サブシステムは、共通サブシステムとビデオ拡張ドライバーから構成されます。この２つのモジュールの働きによって、DOS/Vビデオ拡張機能が実現されます。

共通サブシステムは、BIOS INT10H割り込みのレベルで全角文字の処理を可能にするBIOSに対する拡張部分です。標準のDOS/Vでは、$DISP.SYSと呼ばれるデバイス・ドライバーによって実現されていますが、その他のデバイス・ドライバー、または特定のメモリー・マネージャーに組み込まれたプロテクト・モード・コードで実現されている場合もあります。

共通サブシステムは、基本(従来)テキスト・モード、およびグラフィックス・モードの場合、すべてのINT10Hの処理をそれ自身で、またはROM BIOSを呼び出して行います。一方、拡張テキスト・モードの場合、テキスト、カーソル、およびパレットに関する上位レベルの処理を行い、下位レベルのハードウェアに依存する処理をビデオ拡張ド

ライバーに依頼します。このとき呼び出されるビデオ拡張ドライバーの機能をプリミティブ機能と呼びます。

ビデオ拡張ドライバーは、特定のビデオ・コントローラーのサポート、または特定のビデオ拡張機能の実現を目的として提供されます。その例として、XGA用ビデオ拡張ドライバー、標準800x600用拡張ドライバー、24ドット文字用拡張ドライバーなどが考えられます。この章で述べる仕様は、ビデオ拡張ドライバー固有の機能を定義するのではなく、共通サブシステムとビデオ拡張ドライバーのインターフェースを定義しています。

共通サブシステムの主な役割は次の通りです。

- 基本モードおよびグラフィックス・モードでのすべてのINT10H処理
- ビデオ拡張ドライバーの登録と登録解除
- 論理テキスト・バッファーの管理
- 論理カーソルの管理
- 論理パレットの管理
- BIOS INT10H機能からビデオ拡張ドライバーのプリミティブ機能への変換

ビデオ拡張ドライバーの主な役割は次の通りです。

- 拡張テキスト・モードへの設定と拡張テキスト・モードの解除
- 文字フォント・イメージの表示
- カーソル・イメージの表示と消去（可能であれば、ハードウェア・カーソルをお使いください）
- カラー・パレットの変更
- VRAMのスクロールと塗りつぶし
- フォントの変換
- ビデオ拡張ドライバーの状態の退避または回復

本仕様のバージョン

本仕様のバージョン番号は次の通りです。

メジャー・バージョン番号：　01 (01H)

マイナー・バージョン番号：　20 (14H)

バージョン番号は、次に説明する機能の定義とそのデータ構造を決定付けます。したがって、将来、機能またはデータ構造が拡張または変更されれば、バージョン番号も変わることになります。

以下の説明の中で特にバージョンの明記されていない機能は、バージョン**1.00** 以上で機能します。

共通サブシステムとのインターフェース(INT15H, AX = 501xH)

共通サブシステムとのインターフェースは、INT15H、AX=501xH機能を通して行われます。

AX = 5010H ビデオ拡張情報の照会

<結果>

(ES:BX) = 情報テーブルのアドレス (CF = 0かつAH = 0 の場合に有効)

情報テーブルについては、次の説明を参考にしてください。

(CF) = 0 正常に完了(AH = 0)
= 1 エラーにより未完了(AH = 0以外)

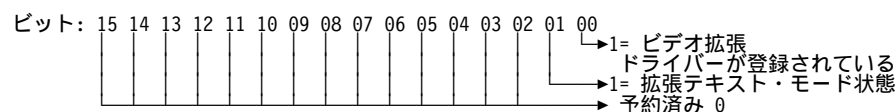
(AH) = 完了コード（以下を参照）

完了コード

00H 正常に完了
86H 機能はサポートされていない

情報テーブル

1バイト00H 共通サブシステムのメジャー・バージョン番号
1バイト01H 共通サブシステムのマイナー・バージョン番号
1ワード02H 拡張ビデオ情報
1ワード04H テキスト・バッファーのセグメント
1ワード06H テキスト・バッファーのサイズ（バイト）
2ワード08H 「ビデオ拡張ドライバーの登録」機能のパラメーター・テーブルへのFARポインター
1ワード0CH ロック・カウンター
2ワード0EH 元のINT10H機能ルーチンのアドレス
1ワード12H フォント・サイズ設定テーブルへのNEARポインター
1ワード14H テキスト密度設定テーブルへのNEARポインター
1ワード16H パレットとオーバー・スキャンの退避領域へのNEARポインター
1バイト18H 現行拡張テキスト・モードのインデックス
1バイト19H 予約済み (0) (バージョン **1.10** 以上)
1ワード1AH 共通サブシステムの機能性 (バージョン **1.10** 以上)
1ワード1CH 共通サブシステム状況ワード (バージョン **1.10** 以上)

拡張ビデオ情報: 1ワードのそれぞれ1ビットが、共通サブシステムまたはビデオ拡張ドライバーのある特定の状態を示しています。

ビット01が1ならば、現行の画面は現行拡張モード・インデックスによって示されたビデオ・モードに設定されています。

テキスト・バッファー:　テキスト・バッファーは、基本テキスト・モードと拡張テキスト・モードで使われます。このバッファーは、共通サブシステムによって、割り振られます。

ビデオ拡張ドライバーは、テキスト・バッファーのサイズが必要とするサイズよりも小さい場合は、エラーを表示して、登録処理を中止するか、またはテキスト・バッファーの不足する拡張テキスト・モードだけを無効な状態にした後に登録を行う必要があります。テキスト・バッファーが不足する拡張テキスト・モードが有効なまま、登録しようとしても、共通サブシステムからビデオ拡張ドライバーに対してエラー・コードが返されます。特定の拡張モードを無効にする方法については、2-18ページの
『AX = 5011H ビデオ拡張ドライバーの登録』の「モード情報」をお読みください。

パラメーター・テーブルへのFARポインター:　このフィールドは、ビデオ拡張ドライバーが登録されている場合のみ有効です（「拡張ビデオ情報」フィールドのビット00が1の場合）。このFARポインターにより、現在登録されているビデオ拡張ドライバーについての種々の情報を参照することができます。

共通サブシステムはビデオ拡張ドライバーを登録する場合に、このフィールドのFARポインターを正規化しない(セグメントおよびオフセットの値を変更しない)ので、このポインターのセグメント部をビデオ拡張ドライバーのデータ構造を含むセグメントの値として使うことができます。

ロック・カウンター:　このカウンターは、現在いくつのプログラムがビデオ拡張ドライバーをロックしているかを示しています。値0は、ビデオ拡張ドライバーをロックしているプログラムがなく、登録解除が自由にできることを意味しています。0でない値の場合は、共通サブシステムや他のシステム・プログラムがビデオ拡張ドライバーをロックしており、登録解除ができないことを意味しています。

共通サブシステムは、画面を拡張テキスト・モードに設定した場合にこのカウンターを増分(+1)し、基本テキスト・モードに戻した場合に減分(−1)します。

元のINT10H機能ルーチンのアドレス:　ビデオ拡張ドライバーによっては、特定の拡張モードに設定するために、ビデオ・レジスターを直接設定する代わりに元のINT10H機能 (通常はROM BIOS機能）を呼び出すことが必要になります。このとき、ビデオ拡張ドライバーはINT10H命令を発行するのではなく、元のINT10H機能ルーチンを呼び出さなければいけません。共通サブシステムは、ブート時に取得した元のINT10H機能ルーチンのアドレスを、このフィールドにコピーします。

フォント・サイズ設定テーブルとテキスト密度設定テーブル:　最初のワードは、このテーブルのエントリーの数を示しています。それに続く２バイトのエントリーは、それぞれのテキスト・モードにおいての現在のフォント・サイズまたはテキスト密度の設定を示しています。モード番号はBIOS画面モード番号であり、モード・テーブル・インデックスは「ビデオ拡張ドライバーの登録」機能（INT15H, AX=5011H)で登録される拡張モード・テーブルのインデックスです。モード番号は、フォント・サイズ設定テ

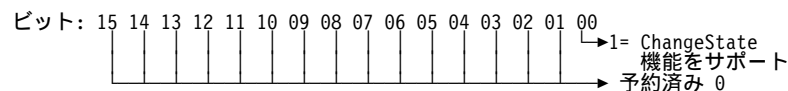
ーブルに対しては03Hまたは73H、テキスト密度テーブルに対しては03H、73H、70H、または71Hです。これらのテーブルは、「DOS/V フォント・サイズの切り替え」機能（INT10H, AH=12H, BL=38H)および「DOS/Vテキスト密度の切り替え」機能（INT10H, AH=12H, BL=39H)で設定されます。これらの機能が呼び出されるまで、モード・テーブル・インデックスのフィールドは、初期値として－１に設定されます。

ワード00H　　以下のエントリーの数
バイト02H,03H モード番号、モード・テーブル・インデックスまたは -1
バイト04H,05H モード番号、モード・テーブル・インデックスまたは -1
. . .

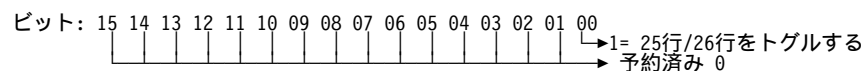
パレットとオーバースキャンの退避領域:　この退避領域の内容は、現在、各論理パレットに設定されているカラー・レジスター番号の値です。ただし、「拡張ビデオ情報」フィールドの「拡張テキスト・モード状態」ビットが1の場合にのみ有効です。

バイト00H　　パレット0
バイト01H　　パレット1
. . .
バイト0FH　　パレット15
バイト10H　　オーバースキャン

現行拡張テキスト・モードのインデックス:　このフィールドは「拡張ビデオ情報」フィールドの「拡張テキスト・モード状態」ビットが1の場合に有効です。このインデックスは、ビデオ拡張ドライバーによって登録された拡張モード・テーブルのインデックスです。

共通サブシステムの機能性:

このフィールドは，共通サブシステムのバージョン番号が**1.10以上**のとき有効です。

ビット00は，標準テキスト・モードが現在，80×25モードと80×26モードのどちらに設定されているかを示します。

共通サブシステム状況ワード:

このフィールドは，共通サブシステムのバージョン番号が**1.10以上**のとき有効です。

AX = 5011H ビデオ拡張ドライバーの登録

<設定>

(ES:BX) = パラメーター・テーブルのアドレス

パラメーター・テーブルについては次の説明をご覧ください。以下のすべてのNEARポインターに関しては、このESの値がセグメント値と仮定されます。

<結果>

(CF) = 0　正常に完了(AH = 0)
= 1　エラーにより未完了(AH = 0以外)

(AH) = 完了コード（以下を参照）

完了コード

00H　正常に完了
01H　既にビデオ拡張ドライバーが登録済み
02H　テキスト・バッファーのサイズが拡張モード・テーブルから計算される値よりも小さい
03H　ビデオ拡張ドライバーの登録拒否(ロック・カウンターが0でない)
86H　機能はサポートされない

パラメーター・テーブル

バイト00H	ビデオ拡張ドライバーのメジャー・バージョン番号
バイト01H	ビデオ拡張ドライバーのマイナー・バージョン番号
ワード02H	プリミティブ機能テーブルへのNEARポインター
ワード04H	拡張モード・テーブルへのNEARポインター
ワード06H	拡張モード・テーブルのエントリー数
ワード08H	ビデオ拡張ドライバー固有文字列へのNEARポインター
ワード0AH	ビデオ拡張ドライバーの機能性 (バージョン **1.10** 以上)

ビデオ拡張ドライバーのバージョン番号:

このフィールドは，ビデオ拡張ドライバーがどのレベルの DOS/Vビデオ拡張仕様に一致しているかを示しています。共通サブシステムや他のシステム・プログラムはこの値から，機能呼び出しのパラメーター構造とビデオ拡張ドライバーがサポートしているプリミティブ機能の種類を判定します。

本仕様のバージョン番号は次のとおりです。

メジャー・バージョン番号:　01 (01H)
マイナー・バージョン番号:　20 (14H)

以下の説明の中で特にバージョンの明記されていない機能は、バージョン**1.00** 以上のビデオ拡張ドライバーで機能します。

プリミティブ機能テーブル

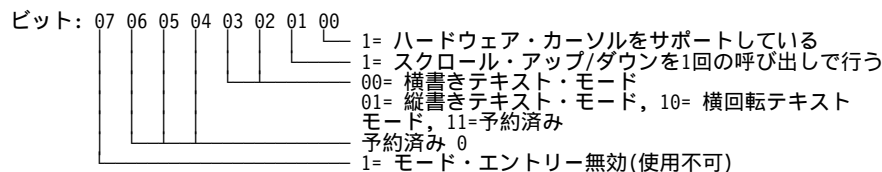
ワード00H　SetToExtVideoMode機能へのNEARポインター
ワード02H　ResetToVGAMode機能へのNEARポインター
ワード04H　WriteSBCSChar機能へのNEARポインター
ワード06H　WriteDBCSChar機能へのNEARポインター
ワード08H　FillRectangle機能へのNEARポインター
ワード0AH　ScrollUp機能へのNEARポインター
ワード0CH　ScrolDown機能へのNEARポインター
ワード0EH　SetCursorShape機能へのNEARポインター
ワード10H　PutCursor機能へのNEARポインター
ワード12H　EraseCursor機能へのNEARポインター
ワード14H　SetPalette機能へのNEARポインター
ワード16H　ChangeFont機能へのNEARポインター
ワード18H　ReturnStateSize機能へのNEARポインター
ワード1AH　SaveState機能へのNEARポインター
ワード1CH　RestoreState機能へのNEARポインター
ワード1EH　ChangeState機能へのNEARポインター (バージョン **1.10** 以上)

拡張モード・テーブル:　テーブルのエントリーはそれぞれ16バイトで、次に示すフォーマットに従っています。

バイト00H　モード番号（以下を参照）
バイト01H　モード情報（以下を参照）
バイト02H　桁数
バイト03H　行数
バイト04H　文字セル幅（半角文字の幅)
バイト05H　文字セルの高さ
バイト06H　文字フォント幅（半角文字の幅)
バイト07H　文字フォントの高さ
ワード08H　水平画面解像度
ワード0AH　垂直画面解像度
バイト0CH　使用する文字フォントの幅（半角文字の幅）(バージョン **1.20** 以上)
バイト0DH　使用する文字フォントの高さ (バージョン **1.20** 以上)
バイト0EH　フォント・インデックス (バージョン **1.20** 以上)
　　bit 7～4:SBCSフォント・インデックス
　　bit 3～0:DBCSフォント・インデックス
バイト0FH　予約済み

このテーブルにおけるエントリーの順序には重要な意味があります。順序番号（拡張モード・テーブルのインデックス）は、対象とする拡張画面モードを指定するために、SetToExtVideoModeまたはその他のいくつかの機能で使用されます。

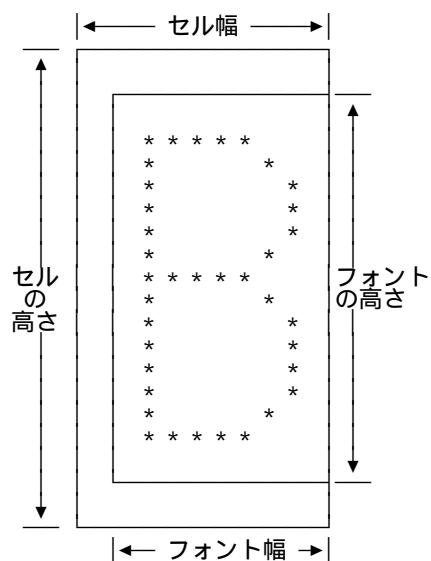
モード番号:　モード番号はBIOS画面モードの番号を指しています。１つのモード番号に対して、複数の拡張モードを用意することができます。ただし、各フィールドがまったく同じである重複したモードは許されません。

モード情報:

ハードウェア・カーソルとは、表示、移動、消去のときに、ビデオ・メモリーに影響せずに、そのイメージがテキスト位置のどこにでもマップできるカーソルのことです。（ハードウェア・カーソルのイメージを保管するために、ビデオ・メモリーの非表示部分が使用されることがありますが、ここではそれを考慮しません。）

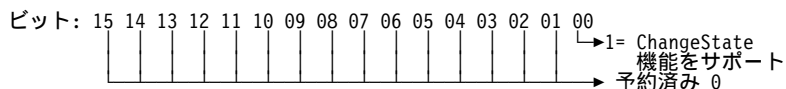
ハードウェア・カーソルがサポートされると、PutCursorとEraseCursorの呼び出し回数が減るため、文字表示や、スクロールの処理時間が短縮されます。ソフトウェア・カーソルが使われると、（ソフトウェア・カーソルとは、ソフトウェアによってカーソル位置のビデオ・メモリーにそのイメージが書き込まれたものです。）PutCursorとEraseCursorは、文字表示とスクロールのたびに呼び出されます。

ビット01が1の場合、ALが0以外でINT10H, AH=06H(スクロール・アップ)、またはINT10H, AH=07H(スクロール・ダウン)機能が呼び出されると、その呼び出しは、それぞれFillRectangleを伴わないScrollUpとScrollDown呼び出しに変換されます。逆に、ビット01が0の場合、それぞれScrollUPとScrollDown呼び出しの後にFillRectanle呼び出しが行われます。ALとして0が指定されたときは、常にFillRectangleだけが呼び出されます。

ビット07が1ならば、共通サブシステムはこのモード・エントリーを無視します。ビデオ拡張ドライバーがなんらかの理由により(たとえば、高品位フォントが利用可能でないなど)、特定の拡張テキスト・モードだけを無効な状態にする必要がある場合は、このビットを1にして登録してください。

セル幅とセルの高さ、およびフォント幅とフォントの高さ:　次の図ではセル幅、セルの高さ、フォント幅、フォントの高さを表しています。周辺のスペースは水平または垂直の罫線、下線のための空間、または文字と文字の間隔を空けるための空間です。

ビデオ拡張ドライバー固有文字列:

この文字列は、ビデオ拡張ドライバーを識別するために使われます。この文字列には、フォーマットの制限はありませんが、最後を0で終わらせてください。

ビデオ拡張ドライバーの機能性:

このフィールドは，ビデオ拡張ドライバーのバージョン番号が**1.10以上**のとき有効です。

AX = 5012H ビデオ拡張ドライバーの登録解除

<結果>

(CF) = 0 正常に完了(AH = 0)
= 1 エラーにより未完了(AH = 0以外)

(AH) = 完了コード（以下を参照）

完了コード

00H 正常に完了
01H ビデオ拡張ドライバーが登録されていない
02H ビデオ拡張ドライバーがロックされている
86H 機能がサポートされていない

注: ビデオ拡張ドライバーが共通サブシステムやその他のシステム・プログラムにロックされていると、ビデオ拡張ドライバーを登録解除できません。他のシステム・プログラムがビデオ拡張ドライバーをロックしていないとしても、現行画面

モードが拡張テキスト・モードだとしたら、ビデオ拡張ドライバーは共通サブシステムにロックされていることになります。この場合、ビデオ拡張ドライバーを登録解除する唯一の方法は、この機能を呼び出す前に一時的に基本画面モード（たとえばモード12H）に切り替えることです。

AX = 5013H ビデオ拡張ドライバーのロックとロック解除

<設定>

(BX) = 1:ロック　-1：ロック解除

<結果>

(CF) = 0 正常に完了(AH = 0)
= 1 エラーにより未完了(AH = 0以外)

(AH) = 完了コード（以下を参照）

完了コード

00H 正常に完了
86H 機能がサポートされていない

注: 共通サブシステムや他のシステム・プログラムがビデオ拡張ドライバーをロックあるいはロック解除するためにこの機能を呼び出すと、ロック・カウンターにBXが加えられます。ロック・カウンターに関しては、「ビデオ拡張情報の照会」の説明をお読みください。

プリミティブ機能

ビデオ拡張ドライバーが、共通サブシステムに登録されると、拡張テキスト・モードの間、共通サブシステムがビデオ拡張ドライバーのプリミティブ機能を呼び出します。プリミティブ機能の役割は、VGAまたは拡張VGAのハードウェア（特にVGAや拡張VGAのレジスターやAPAメモリー）を管理することだけです。プリミティブ機能はテキスト・バッファーやBIOSデータ・エリアを変更する必要はまったくありません。テキスト・バッファーや BIOS データエリアを管理するのは共通サブシステムの役割です。

[規約]

共通サブシステムからビデオ拡張ドライバーに渡されるすべてのパラメーターは 8086 汎用レジスターに格納されます。ビデオ拡張ドライバーは、戻る場合にすべてのレジスターの内容を元どおりの状態にしなければなりません。言い換えれば、ビデオ拡張ドライバーで変更するレジスターを退避または回復するのはビデオ拡張ドライバーの役目です。すべてのパラメーターは、それぞれのプリミティブ機能が呼び出される前に、妥当性がチェックされますので、ビデオ拡張ドライバーはチェックする必要はありません。

ビデオ拡張ドライバーは、プリミティブ機能から戻るときに呼び出し元に対してRET FARで戻ります。

SetToExtVideoMode

<設定>

(AL) = 拡張モード・テーブルのインデックス

(AH) = 予約済み　（常に0）

この機能が呼び出されてから次にResetToVGAModeが呼び出されるまで、画面は拡張テキスト・モードになります。ビデオ拡張ドライバーは、VGAモードから、(AL)で指定された拡張テキスト・モードに対応するモードに、ビデオ・コントローラーを切り替えます。ドライバーは直接I/Oレジスターを操作するか、元のINT10H機能を呼び出すことによって、モードを設定します。また、ドライバーは必要なすべての変数を、このとき初期化します。

ビデオ拡張ドライバーは、パレットなどの色情報もこの機能の中で設定しなければなりません。共通サブシステムは、パレットの内容を初期値から変更する必要があるときまで、SetPaletteを呼び出しません。

戻りの際にはカーソルを「非表示」の状態に設定してください。カーソルはPutCursor機能またはEraseCursor機能で表示または消去されます。カーソルの形はSetCursorShape 機能で設定または変更されます。

ResetToVGAMode

<設定>

なし

ビデオ拡張ドライバーは、現行拡張モードからVGAモードへの切り替えだけを行います。特定のVGAモードに設定するためにすべてのVGAレジスターに設定する必要はありません。VGAレジスターの設定はこの機能から制御が戻った後、共通サブシステムが行います。

ビデオ拡張ドライバーがSetToExtVideoMode機能でVGAモードから特定の拡張モードに切り替える必要がない場合、つまり標準VGAモードでそのビデオ拡張機能を実現している場合(たとえば、VGAグラフィックス・モード12Hを使った高密度テキスト・モードなど)は、この機能はRET FAR命令のみを持つことになります。

WriteSBCSChar

<設定>

(AL) = 属性 1

(AH) = 属性 2 (3バイト属性モードの場合のみ有効)

(CH) = 0

(CL) = 半角文字コード

(DX) = 文字を書く位置(行番号、桁番号)

WriteDBCSChar

<設定>

(AL) = 全角第１バイトの属性1

(AH) = 全角第１バイトの属性2 (3バイト属性モードの場合のみ有効)

(BL) = 全角第２バイトの属性1

(BH) = 全角第２バイトの属性2 (3バイト属性モードの場合のみ有効)

(CX) = 全角文字コード(シフトJISコード)

(DX) = 文字を書く(行番号、桁番号)

全角文字の第１と第２の属性が等しくない場合は、この機能は左右の文字イメージを別々に表示しなければなりません。たとえば左半分（第１）が赤、右半分（第２）が青で指定されている場合は、左のイメージは赤、右のイメージは青で描かれなければなりません。

この機能は、行の最右端列に全角文字を書き込むために呼び出されることはありません。アプリケーションが全角文字を最右端列に書き込もうとすると、共通サブシステムは最右端列および次行の最左端列(次行があれば)に半角スペースを書き込むようにプリミティブ機能を呼び出します。（一方、テキスト・バッファーはアプリケーションによって指定されたとおりに書き込まれます。）

FillRectangle

<設定>

(AL) = 矩形領域を塗りつぶす色
上位４ビット = 前景色
下位４ビット = 背景色

(CX) = 塗りつぶす矩形領域の左上の隅（行番号、桁番号）

(DX) = 塗りつぶす矩形領域の右下の隅（行番号、桁番号）

この機能は BIOS 機能 INT10H, AH=06H/07H, AL=00H (全ウィンドウ・スクロール)が呼び出された場合に、または部分的スクロール(ScrollUpおよびScrollDown)の後にブランク行を塗りつぶす場合に呼び出されます。

拡張テキスト・モードが特定のグラフィック・モードを使ってエミュレートされている場合は、APAメモリーはALレジスターの下位４ビットの色によって塗りつぶされなけ

ればなりません。ALレジスターの上位４ビットは、拡張テキスト・モードが真のテキスト・モードを使ってエミュレートする場合にのみ使われます。

ScrollUp

<設定>

(AL) = 上方向にスクロールする行数 (ALは0以外)

(AH) = ブランク行を塗りつぶす色（「モード情報」のビット01が1の場合のみ有効。AHの内容はFillRectangleのALの内容と同じ）

(CX) = スクロール領域の左上の隅（行番号、桁番号）

(DX) = スクロール領域の右下の隅（行番号、桁番号）

「モード情報」のビット01が0の場合、この機能は指定された領域を上にスクロールさせるだけで、スクロールの後にブランク行を塗りつぶす必要はありません。ブランク行を塗りつぶすのは、FillRectangle機能が行います。「モード情報」のビット01が1の場合は、この機能は指定された領域を上にスクロールすると同時にブランク行を塗りつぶす必要があります。

ハードウェア・スクロール方式を用いるのかどうかは、ビデオ拡張ドライバーが決定します。

BIOS機能INT10H, AH=06H, AL=00H (全矩形領域の塗りつぶし）はFillRectangle機能に変換されますので、ALが0になることはありません。

ScrollDown

<設定>

(AL) = 下方向にスクロールする行数 (ALは0以外)

(AH) = ブランク行を塗りつぶす色（「モード情報」のビット01が1の場合のみ有効。AHの内容はFillRectangleのALの内容と同じ）

(CX) = スクロール領域の左上の隅（行番号、桁番号）

(DX) = スクロール領域の右下の隅（行番号、桁番号）

「モード情報」のビット01が0の場合、この機能は指定された領域を下にスクロールさせるだけで、スクロールの後にブランク行を塗りつぶす必要はありません。ブランク行を塗りつぶすのは、FillRectangle機能が行います。「モード情報」のビット01が1の場合は、この機能は指定された領域を下にスクロールすると同時にブランク行を塗りつぶす必要があります。

ハードウェア・スクロール方式を用いるのかどうかは、ビデオ拡張ドライバーが決定します。

BIOS機能INT10H, AH=07H, AL=00H (全矩形領域の塗りつぶし）はFillRectangle機能に変換されますので、ALが0になることはありません。

SetCursorShape

<設定>

(CH) = 文字ボックス内のカーソル開始位置（物理値）

(CL) = 文字ボックス内のカーソル終了位置（物理値）

カーソル開始位置または終了位置はCGA文字ボックス(8x8)に対するものではなく、ビデオ拡張ドライバーによって用意された拡張モード・テーブルに示されている文字ボックスに対するものです。CGA文字マトリックスから物理文字マトリックスへの変換は、共通サブシステムによってこの機能呼び出しの前に行われます。

この機能は次の場合に呼び出されます。

- モード設定の後に最初にカーソルが表示される場合
- カーソルの形がBIOS機能(INT10H, AH=01H)を通じて変更される場合

この機能はカーソル表示の現行の状態(「表示」または「非表示」)を変更してはいけません。つまり、この機能が呼び出される前にカーソル表示の現行の状態が「表示」状態である場合は、戻るときにも「表示」状態のままでなければなりません。「非表示」の場合も同じです。

PutCursor

<設定>

(AL) = カーソルの色

(DX) = カーソルの位置（行番号、桁番号）

この機能は次の場合に呼び出されます。

- モード設定の後に、最初にカーソルが表示される場合
- カーソルが移動した場合
- カーソルが消去された後に再表示された場合

これに加えて、ハードウェア・カーソルが使用不可能の場合、この機能はEraseCursorと共にソフトウェアによってカーソルをエミュレートすることが必要なときはいつでも呼び出されます。

カーソルの色はテキスト・カーソルをエミュレートするためには、カーソル位置の文字の前景色と同じであることが必要ですが、ハードウェア・カーソルが使用できない場合は、このような正確なエミュレーションは困難です。このような場合は、たとえば、ビデオ拡張ドライバーは0xFF値でカーソル位置のAPAメモリーをXORすることによってエミュレートします。

EraseCursor

<設定>

なし

この機能はカーソルが消去される場合に呼び出されます。これに加えて、ハードウェア・カーソル機能が使用不可能の場合、この機能はPutCursorと共にソフトウェアによってカーソルをエミュレートすることが必要なときはいつでも呼び出されます。

SetPalette

<設定>

(AL) = パレット番号 (0 - 15)、オーバースキャン (16)

(AH) = カラー・レジスター番号

この機能は、BIOS機能のINT10H, AH=10H, AL = 00H / 01H / 02H が呼び出された場合に呼び出されます。ビデオ拡張ドライバーはAHレジスターのカラー情報をハードウェアに設定します。ビデオ拡張ドライバーが使用するビデオ・コントローラーが、VGAと同じカラー体系を取り入れている場合は、直接パレットにカラー・レジスター値を設定するか、またはBIOS機能の INT10H、AH=10H、AL=00Hと01H を呼び出してパレットにカラー・レジスター値を設定するだけです（直接割り込み命令を発行せずに、元のINT10H機能ルーチンを呼び出してください)。特別のカラー体系を取り入れている場合は、ビデオ拡張ドライバーはそれ自身でカラー・レジスター番号からそのハードウェアーの固有のカラー値に変換してハードウェアーに設定します。VGAのパレット設定、およびカラー・レジスター情報がその変換に必要ならば、ビデオ拡張ドライバーは、その導入時にモードを03HにしてBIOS機能INT10H, AH=10H, AL=09H , およびAL=17Hを呼び出して、すべてのパレットとオーバースキャン値を読み込んでおく必要があります。

ChangeFont

<設定>

(AL) = 半角文字コード・ポイント

(BH) = 文字フォントの幅

(BL) = 文字フォントの高さ

(ES:SI) = フォント・パターンのアドレス

この機能は、「ユーザーの１バイト文字セットの設定」機能(INT10H, AX=1100H)が呼び出された場合に呼び出されます。ビデオ拡張ドライバーは、その文字生成バッファー内の対応する領域に指定されたフォント・パターンをロードします。指定された文字サイズ（幅と高さ）が現行モードに合わない場合に、フォント・パターンを必要なサイ

ズに拡張するかどうかはビデオ拡張ドライバーによります。ビデオ拡張ドライバーは、バッファーへのフォント・パターンのロードだけを行い、新しいフォントを使っての画面の再表示は行ってはいけません。画面の再表示は、共通サブシステムから呼ばれたWriteSBCSCharが行います。

フォント・スライスは、ES:DIで指定されたバッファーにすきまなく格納されています。隣接する２つのフォント・スライス・イメージの間にギャップはありません。(フォント・パターンのビット列は、バイト境界とは無関係に連続して格納されます。)

ReturnStateSize

<設定>

(AL) = 要求ステート (以下を参照)

<結果>

(CX) = バッファー・サイズ

SaveState

<設定>

(AL) = 要求ステート (以下を参照)

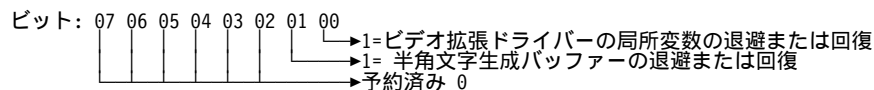
(ES:BX) = ステートを退避するための呼び出し元のバッファーへのFARポインター

RestoreState

<設定>

(AL) = 要求ステート (以下を参照)

(ES:BX) = ステートを回復するための呼び出し元のバッファーへのFARポインター

要求ステート

ビデオ拡張ドライバーは、局所変数および半角文字生成バッファーを呼び出し元のバッファーに退避、またはバッファーから回復します。半角文字生成バッファーの内容は、ビデオ拡張ドライバーのメモリー・スペース、またはビデオ・メモリー・スペースに格納されています。半角文字生成バッファー内容を退避する場合は、ビデオ拡張ドライバーは退避先のスペースを節約するために、すきまをあけずにデータを格納してください。隣接する２つの文字フォント・イメージの間、および２つのフォント・スライス・イメージの間にギャップを入れないでください。

ChangeState

<設定>

(AX) = 状況変更ビット（以下を参照）

(BL) = 0: 状況変更の実行

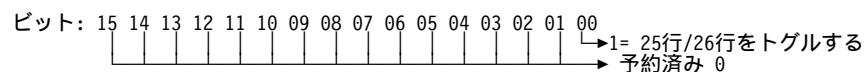
= 1: 機能性のチェック

<結果>

(AL) = 機能性（（BL）=1 の場合のみ有効）

0: (AX)で指定された状況変更をサポートしている

–1: (AX)で指定された状況変更をサポートしていない

状況変更ビット:

この状況変更ビットで複数の状況変更が指定されることはありません。複数の状況変更が必要な場合は、その回数だけChangeStateが呼び出されます。

ビット00 ビデオ拡張ドライバーは、拡張モード・テーブル内のモード03Hと73Hに対応するエントリーのうち、80桁×25行、または80桁×26行であるものを指定しなおします。それ以外のエントリーは変更しません。この設定変更は、INT 10H, AX=1D03Hの機能呼び出しを実現するためのものです。

ビデオ拡張ドライバーは、（BL）=0のとき(AX)で指定された状況変更を実行します。（BL）=1のときは(AX)で指定された状況変更をサポートしているかどうかを（AL）に返します。

ビデオ拡張ドライバーのプログラミング上の考慮点

ビデオ拡張ドライバーが参照できるデータ

ビデオ拡張ドライバーは、すべての自分の局所データと、プリミティブ機能の呼び出しで渡される8086汎用レジスターの内容を参照できますが、BIOSデータ・エリアを直接参照してはいけません。これは、BIOSデータ・エリアを操作することなく、共通サブシステムを含むすべてのプリミティブ機能の呼び出し手がビデオ拡張ドライバーを制御できるようにするためです。

スタック使用法

プリミティブ機能は、主に共通サブシステムによって、INT10Hの処理の中から呼び出されます。共通サブシステムは、ビデオ拡張ドライバーのためにスタック領域は割り振りません。言い換えれば、ビデオ拡張ドライバーは、(独自にスタック・スペースを割り振らない限り、)アプリケーションのスタック・スペースを使うことになります。このため、プリミティブ機能の処理ルーチンで、必要以上のスタック領域を使わないようにしてください。または、独自のスタック・スペースを割り振る場合は、再帰的に呼び出された場合も、暴走したり、ハングアップしたりしないように注意してプログラムしてください。

文字フォント・パターン

フォント・パターンの取得: ビデオ拡張ドライバーは、文字を表示するために文字フォント・パターンが必要です。フォント・パターンはフォント・サブシステムを呼び出すことによって得られます。そして、もし必要ならば(*1)、フォント・パターンを文字セル・サイズに合うように拡張してください。ビデオ拡張ドライバーは、サポートするコード・ページに依存しないようにするため、独自にフォント・パターンを用意しないでください。

(*1): ╓ または ╥ のようなボックス文字のフォント・パターン(半角文字コード: 01H, 02H, 03H, 04H, 05H, 06H, 10H, 14H, 15H, 16H, 17H, 19H, 1AH, 1DH)は、ビデオ拡張ドライバーが標準半角フォント(8x16, 8x19, 12x24)のサイズ以上のフォント・パターンを必要とする場合のみ文字セル・サイズいっぱいに拡張してください。

半角フォント・パターンの変更: 標準のDOS/Vビデオ・サブシステムは、アプリケーションが指定した半角文字フォントを、半角文字生成バッファーにロードする機能(INT10H, AX=1100H)をサポートしています。この機能を使う既存のアプリケーションは、16ドット・テキスト・モード用のフォント・パターンを提供しています。高品位テキスト・モードにおいて、品質を維持しつつ拡張することが可能ならば、フォント・パターンを対応するサイズに拡張してください。高品位テキスト・モードにおいて、文字フォントを拡張するかどうかは、ビデオ拡張ドライバーをどのように実現するかに依存します。

ビデオ拡張プロファイル

ビデオ拡張ドライバー、テキスト・モード設定プログラム(DSPX.EXE)、あるいはCHEV.EXEなどのDOS/Vプログラムは、ビデオ拡張機能に関するいくつかの情報をプロファイルとして共有します。プロファイルは、導入ディスケット（導入元）の中に各ビデオ拡張ドライバーごとに１つのプロファイル（そのファイル名は、ドライバーのファイル名の拡張子を.PROに変えたもの）が用意されます。導入時に、選択されたビデオ拡張ドライバーに対応するプロファイルがDSPX.PROというファイル名でシステムにコピーされます。

プロファイルのフォーマット

エントリー

プロファイルは、複数個のエントリーから構成されるバイナリーのファイルです。各エントリーは、**Length**(2バイト), **Entry-ID**(16バイト)、および**Content**(可変長)です。

Length： そのエントリーが何バイトから構成されるかを示します(Lengthを含む)。Lengthが0の場合は、そこがプロファイルの終了であることを示します。

Entry-ID： そのエントリーがどの種類の情報かを示す文字列です。(‘VideoCardInfo’, ‘DriverFileName’, ‘VideoModeTable’, ‘DSPXInfo’, ‘DBCSVideoMode’, ‘SBCSVideoMode’,
または‘OptionTable’)

Content： そのエントリーの情報そのものです。

フォーマット

プロファイルは次のフォーマットを持ちます。

```
Length, "VideoCardInfo   ", サポートするビデオ・コントローラーの
                             名前(30バイト以下)
Length, "DriverFileName  ", ビデオ拡張ドライバーのファイル名(12バイト)
Length, "VideoModeTable  ", ビデオ拡張仕様のバージョン(1ワード)
                             エントリー数(1ワード),
                             ModeTable(0),
                             ModeTable(1),
                                 :
                             ModeTable(m),
Length, "DSPXInfo        ", DSPX情報(1ワード)
                                  bit0 = 0(簡易表示)
                                         1(詳細表示),
                                  bit2,1 = 00(環境43行(注4を参照))
                                           01(英語環境50行(注4を参照))
                                           10(予約済み)
                                           11(予約済み)
Length, "DBCSVideoMode   ", モード番号(1バイト),
                             予約済み(1バイト),
                             画面の桁数(1バイト),
                             画面の行数(1バイト),
Length, "SBCSVideoMode   ", モード番号(1バイト),
                             予約済み(1バイト),
                             画面の桁数(1バイト),
                             画面の行数(1バイト),
Length, "OptionTable     ", オプションの数(1ワード),
                             OptionSyntax(0), 00h, OptionHelp(0), 00h,
                             OptionSyntax(1), 00h, OptionHelp(1), 00h,
                                 :
                             OptionSyntax(n), 00h, OptionHelp(n), 00h,
Length(0)
```

注:

1. **VideoCardInfo**は、サポートされているビデオ・コントローラーをビデオ拡張ドライバー選択パネルで表示するために、導入プログラム(INSTALL.EXE)によって使われます。

2. **DriverFileName**は、ビデオ拡張ドライバーのファイル名を認識するために、導入プログラムによって使われます。対応するドライバーのファイルは導入先システムにコピーされます。このファイル名を含んでいるステートメントが、そのファイルを自動始動するために、AUTOEXEC.BATファイルに書き込まれます。

3. **VideoModeTable**は、ビデオ拡張ドライバーによってその起動時に読まれ、その後にテキスト・モード設定の変更が行われるときに、DSPX.EXEによって更新されます。

4. **DSPXInfo**のビット0が0の場合は、テキスト・モード設定プログラム(DSPX.EXE)は初心者用の簡易表示の初期画面を表示します。ビット0が1の場合は、熟練者用の詳細表示の初期画面を表示します。ビット2, 1はCHEV.EXEコマンドで英語環境にある場合に使用されます。ビット2, 1が00の場合は、英語環境に

おける縦長テキスト・モードが80x43であることを示し、01の場合は、80x50であることを示します。それ以外の値10と11は予約済みです。

5. **DBCSDisplayMode**と**SBCSDisplayMode**は、CHEV.EXEコマンドが英語環境または日本語環境に切り替える場合に、画面モードを設定するために参照します。

6. **OptionTable**は、使用可能なオプションを表示し、ユーザーにオプションを選択させるために、導入プログラムによって使われます。選択されたオプションは、ドライバーのファイル名と共にAUTOEXEC.BATファイルに書き込まれます。

7. **OptionSyntax**は、ドライバーのためのコマンド・パラメーターの構文です。フォーマットは次の通りです。（長さは64バイト以下）

   (タイプA) オプション名 = オプション(1) / オプション(2) / ... / オプション(k)

   (タイプB) オプション名 = [桁数]

   導入プログラムはOptionSyntaxを読みます。そして、ユーザーにタイプAでのオプション(i)を選択させるか、またはタイプBでの値(桁数はオプションを指定する数字の最大桁数(または最大文字数)を指します）を指定させます。その後、AUTOEXEC.BATに、タイプAでの“**DSPXxxxx オプション名 = オプション(i)** ”、またはタイプBでの“**DSPXxxxx オプション名 = nnn** ”を入れます。

8. **OptionHelp**は、ユーザーがオプションを選択する画面でヘルプ・キー(**F1**キー)を押した場合に、導入プログラムによって表示されるストリングです。ストリングの長さは、512バイト以下です。画面に表示される場合に、改行が必要な箇所には、改行(OAH)を入れます。

9. **ModeTable**のフォーマットは次の通りです。

   バイト　省略時値(1)または省略時値以外(0)
   バイト　モード番号
   バイト　モード情報(ビット7 = 妥当性,ビット3, 2 = ビデオ・モード・タイプ）
   バイト　桁数
   バイト　行数
   バイト　文字ボックスのセルの幅
   バイト　文字ボックスのセルの高さ
   バイト　フォントのセルの幅
   バイト　フォントのセル高さ
   ワード　画面の水平解像度
   ワード　画面の垂直解像度
   バイト　使用するフォントのセルの幅
   バイト　使用するフォントのセルの高さ
   バイト　フォント・インディクス
   bit7～4:半角フォント・インデックス
   bit3～0:全角フォント・インデックス
   バイト　予約済み

テーブルの最初のバイトが1の場合、ビデオ拡張ドライバーは、モードの定義 (標準(80x25)テキスト・モードまたは可変密度テキスト・モード)に従って「DOS/Vフォント・サイズの切り替え」機能 (INT10H, AH=12H, BL=38H)、または「DOS/V テキス

ト密度の切り替え」機能 (INT10H, AH=12H, BL=39H)のどちらかを実行することによって、システムの省略時のモードを設定します。最初のバイトに続くアイテムは、ビデオ拡張ドライバーのパラメーター　の拡張モード・テーブルと同じです。

ビデオ・モード・タイプ（モード情報のビット3,2)が00のとき、テキスト・モードは通常（横書き）テキスト・モードです。ビット3,2が01のとき、テキスト・モードは縦書きテキスト・モードです。ビット3,2が10のとき、テキスト・モードは横回転テキスト・モードです。

使用するフォントのセル幅および高さは、ビデオ拡張ドライバーがフォント・サブシステム ($FONT.SYSなど)から取得するフォントのセル幅および高さです。

フォント・インデックス（バイト）は、どのフォントがテキスト・モードに割り当てられているかを指定します。このバイトの上位4bitは、半角フォントのインデックスを保持し、下位4bitは、全角フォントのインデックスを保持します。これらのインデックスは、DOSV.INIファイルに定義されているフォントのインデックスに対応しています。

プロファイルの例

```
 30, "VideoCardInfo   ", "SuperVGA (800x600)",
 30, "DriverFileName  ", "DSPXSVGA.EXE",
192, "VideoModeTable  ", 0001h, 10,
                         1, 03h, 00h, 80, 33, 8, 18, 8, 16, 800, 600, 0, 0,
                         1, 73h, 00h, 80, 33, 8, 18, 8, 16, 800, 600, 0, 0,
                         ...                  :
                         0, 70h, 00h, 80, 50, 8, 12, 8, 12, 800, 600, 0, 0,
 20, "DSPXInfo        ", 0,
 22, "DBCSVideoMode   ", 03h, 0, 80, 25,
 22, "SBCSVideoMode   ", 03h, 0, 80, 25,
100, "OptionTable     ", 2,
                         "HS=ON/OFF/LC", 0, "スクロール法を指定します。
                          ON: ハードウェア・スクロール、
                          OFF: ソフトウェア・スクロール
                          LC: ライン・コンペア・レジスターを用いた
                          ハードウェア・スクロール", 0,
                          "MODE=[2]", 0,
                          "SuperVGAの800x600 16 色を表示する
                          ビデオ・モード番号を16進数で指定します。", 0,
  0
```

第3章　プリンター・サブシステムの拡張機能

この章では、DOS/Vのプリンター・サブシステム(共通サブシステムおよびプリンター・ドライバー)に対してオプションとして定義されている付加機能と、それらを使用する上での考慮点について説明します。これらの拡張された機能には、DOS/V上でプリンター・ドライバーをダイナミックに切り換えたり、登録解除したりするための諸機能が含まれています。

プリンター・ドライバー・インターフェース

プリンター・ドライバー・インターフェースは、プリンター・ドライバー、またはそれに関連するシステム・プログラムを開発するためのものです。アプリケーション・プログラムを開発する場合は、ここで説明するプリンター・ドライバー・インターフェースではなく、必ず、BIOS INT 17H機能を使ってください。

拡張されたプリンター・サブシステムの構成

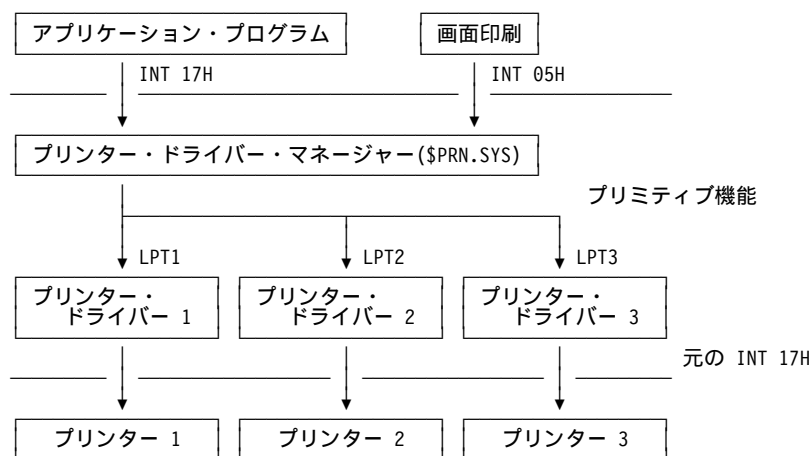



図　3-1. プリンター・サブシステムの構成

プリンター・サブシステムは、プリンター・ドライバー・マネージャーと各種プリンター・ドライバーから構成されます。

プリンター・ドライバー・マネージャーは、BIOS INT 17H 割り込みと INT 05H 割り込みを監視します。BIOS INT 17H が呼び出されると、指定されたプリンター・ポートに登録されているプリンター・ドライバーのプリミティブ機能"プリンターI/O"を呼び出します。また、BIOS INT 05H が呼び出されると、LPT1（プリンター・ポート0）に登録されているプリンター・ドライバーのプリミティブ機能"画面印刷"を呼び出します。いずれの場合もプリンター・ドライバーが登録されていなければ、ROM BIOSの割り込みルーチンが呼び出されます。その他の機能は次のとおりです。

- プリンター・ドライバーの登録情報の提供（INT 15H，AX=5020H）
- プリンター・ドライバーの登録（INT 15H，AX=5021H）
- プリンター・ドライバーの登録解除（INT 15H，AX=5022H）
- プリンター・ドライバー機能の有効/無効の切り換え（INT 15H，AX=5023H）
- プリンター・モードの変更（INT 15H，AX=5024H）

プリンター・ドライバーはTSR（終了後常駐）プログラムで、起動されると自分自身を指定されたプリンター・ポートに登録（INT 15H，AX=5021H）し、メモリーに常駐します。そしてプリンター・ドライバー・マネージャーが呼び出したプリミティブ機能を実行します。プリンター・ドライバーが常駐解除のパラメーター（/R）をつけて起動されると、自分自身を登録解除（INT 15H，AX=5022H）し、割り当てられていたメモリーを解放します。

共通サブシステムとのインターフェース(INT15H, AX = 502xH)

共通サブシステムとのインターフェースは、INT15H、AX=502xH機能を通して行われます。

プリンター・ドライバー情報の取得

＜設定＞

AX = 5020H

＜結果＞

(ES:BX) プリンター・ドライバー情報テーブルへのポインター
(CF)
　= 0　正常に完了（AH=0の場合）
　= 1　エラーにより未完了（AH=0でない場合）
(AH)　完了コード（下記参照）

完了コード:

返されるコードは、以下のとおりです。

00H　正常に完了
86H　機能はサポートされていない

プリンター・ドライバー情報テーブル:

DWORD (0)　元のINT 17Hエントリー
DWORD (4)　元のINT 05Hエントリー
DWORD (8)　LPT1情報
DWORD (12) LPT2情報
DWORD (16) LPT3情報

LPTx情報:

ビット 7-0: プリンターID

ビット 6-0: 論理プリンターID
00H: プリンター・ドライバーは登録されていません
01H: IBM 5575/77プリンター
02H: ESC/P J84プリンター
その他: 予約済み
ビット 7: エミュレーション・ビット
OFF: 接続されているのはDBCSプリンター（エミュレートされていない）
ON: プリンター・ドライバーによりエミュレートされている

ビット 8: ドライバー状況

OFF: プリンター・ドライバーが機能していない（パス・スルー・モード）
ON: プリンター・ドライバーが機能している

ビット 15-9: 予約済み

ビット 31-16: プリンター・ドライバーのセグメント

注:

1. この機能は、どのプリンター・ドライバーがシステムに導入されているかを知る必要があるプログラムとプリンター・ドライバーによって呼び出されます。
2. 「元のINT 17Hエントリー」は、登録されているプリンター・ドライバーがプリンターと通信するときにプリンター・ドライバーによって呼び出されます。
3. 「元のINT 05Hエントリー」は、画面印刷がプリンター・ドライバーでなく「元のINT05Hルーチン」によって行われる必要がある場合に、プリンター・ドライバーによって呼び出されます。
4. ビット8がOFFの場合、プリンター・ドライバー・マネージャーはデータをROM BIOSに直接送ります。このビットがONの場合、プリンター・ドライバー・マネージャーはデータを登録されたプリンター・ドライバーに送ります。
5. 「プリンター・ドライバーのセグメント・アドレス」（ビット31-16）は、登録解除のときにメモリー・ブロックを解放するために、プリンター・ドライバーによって参照されます。

プリンター・ドライバーの登録

＜設定＞

AX = 5021H
CL = プリンターID
DX = プリンター・ポート (0, 1, 2)
ES:BX = 機能テーブルへのFARポインター

プリンターID:

「プリンター・ドライバー情報の獲得（INT 15H，AX=5020H）」を参照してください。

機能テーブル:

（各機能のセグメント・アドレスはES値と同じです）

WORD(0)　PrinterIO機能のエントリー・ポイントへのNEARポインター
WORD(2)　PrintScreen機能のエントリー・ポイントへのNEARポインター
WORD(4)　ChangePrinterMode機能のエントリー・ポイントへのNEARポインター

＜結果＞

(CF)　= 0　正常に完了（AH=0の場合）
　　= 1　エラーにより未完了（AH=0でない場合）
(AH)　= 完了コード（下記参照）

完了コード:

返されるコードは、以下のとおりです。

00H　正常に完了
01H　無効なパラメーター
02H　プリンター・ポートにはすでにプリンター・ドライバーがロードされている
86H　機能はサポートされていない

注:

1. この機能は、プリンター・ドライバーがロードされるときに呼び出されます。
2. プリンター・ポートが0（LPT1）の場合、$PRN.SYSのデバイス名は、IBM DOS J5.0/Vとの互換性を保つために変更されます。（IBM 5575/77には$IBMAUSR、ESC/P J84には$IBMAESP、その他のプリンター・ドライバーにはNULL）
3. 機能テーブルのPrinterIO機能は、プリンターI/Oリクエスト（INT 17H）が発行されたときに、プリンター・ドライバー・マネージャーによって呼び出されます。
4. 機能テーブルのPrintScreen機能は、画面印刷割り込み（INT 05H）が発行されたときに、プリンター・ドライバー・マネージャーによって呼び出されます。
5. 機能テーブルのChangePrinterMode機能は、プリンター・モードの変更（INT 15H，AX=5024H）が発行されたときに、プリンター・ドライバー・マネージャーによって呼び出されます。

プリンター・ドライバーの登録解除

<設定>

AX = 5022H
DX = プリンター・ポート (0, 1, 2)

<結果>

(CF) = 0　正常に完了（AH=0の場合）
= 1　エラーにより未完了（AH=0でない場合）
(AH) = 完了コード（下記参照）

完了コード:

返されるコードは、以下のとおりです。

00H 正常に完了
01H 無効なパラメーター
02H プリンター・ポートにプリンター・ドライバーがロードされていない
86H 機能はサポートされていない

注:

1. この機能は、プリンター・ドライバーが常駐しなくなるときに、そのプリンター・ドライバーによって呼び出されます。
2. プリンター・ポートが0（LPT1）の場合、$PRN.SYSのデバイス名は、NULL（8 dup（0））に変更されます。

プリンター・ドライバー機能の有効/無効の切り換え

<設定>

AX = 5023H
BL = 0: プリンター・ドライバー機能を無効にする
= 1: プリンター・ドライバー機能を有効にする
DX = プリンター・ポート (0, 1, 2)

<結果>

(CF) = 0　正常に完了（AH=0の場合）
= 1　エラーにより未完了（AH=0でない場合）
(AH) = 完了コード（下記参照）

完了コード:

返されるコードは、以下のとおりです。

00H 正常に完了
01H 無効なパラメーター
02H プリンター・ポートのプリンター・ドライバー機能はすでに有効/無効になっている
86H 機能はサポートされていない

プリンター・モードの変更

<設定>

AX = 5024H
BL = 0: SBCSモード
= 1: DBCSモード
DX = プリンター・ポート (0, 1, 2)
= −1（すべてのプリンター・ポート）

<結果>

(CF) = 0　正常に完了（AH=0の場合）
= 1　エラーにより未完了（AH=0でない場合）
(AH) = 完了コード（下記参照）

完了コード:

返されるコードは、以下のとおりです。

00H　正常に完了
01H　無効なパラメーター
02H　プリンターの準備ができていない
86H　機能はサポートされていない

注:　DXに−1を指定した場合、完了コードとして02Hが返されることはありません。つまり、準備ができていないプリンターがあってもエラーにはなりません。

プリミティブ機能

プリンター・ドライバーがプリンター・ドライバー・マネージャーに登録されると、次に登録解除されるまでプリンター・ドライバー・マネージャーはプリンター・ドライバーのプリミティブ機能を呼び出します。

[規約]

プリンター・ドライバー・マネージャーからプリンター・ドライバーに渡されるすべてのパラメーターは 8086汎用レジスターに格納されます。プリンター・ドライバーは、戻る場合に、戻り値を格納するレジスターを除くすべてのレジスターの内容を、元どおりの状態にしなければばなりません。言い換えれば、プリンター・ドライバーで変更するレジスターを退避または回復するのはプリンター・ドライバーの役目です。すべてのパラメーターは、それぞれのプリミティブ機能が呼び出される前に、妥当性がチェックされますので、プリンター・ドライバーはチェックする必要はありません。

プリンター・ドライバーは、プリミティブ機能から戻るときに呼び出し元に対して RET FARで戻ります。

PrinterIO

文字の印刷

＜設定＞

(AH)　=　00H
(AL)　=　データ
(DX)　=　プリンター・ポート（0，1，2）

＜結果＞

(AH)　=　状況（BIOS INT 17H機能におけるプリンターの状況に準じます）

プリンター・ポートの初期設定

＜設定＞

(AH)　=　01H
(DX)　=　プリンター・ポート（0，1，2）

＜結果＞

(AH)　=　状況（BIOS INT 17H機能におけるプリンターの状況に準じます）

状況の読み取り

＜設定＞

(AH)　=　02H
(DX)　=　プリンター・ポート（0，1，2）

＜結果＞

(AH)　=　状況（BIOS INT 17H機能におけるプリンターの状況に準じます）

プリンター・ドライバーは、プリンター・ポートを直接アクセスして、または元のINT 17Hのエントリーを呼び出すことによって上記の機能を実現します。

PrintScreen

＜設定＞
なし

＜結果＞
なし

プリンター・ドライバーは、画面印刷を自ら行うときは、その処理の前に、バイト領域0050:0000（16進）を1にし、処理が終ったら、バイト領域 0050:0000（16進）を0にします。元のINT 05Hのエントリーを呼び出して画面印刷を行うときは、バイト領域0050:0000（16進）を変更せずに（0のまま）、元のINT 05Hのエントリーを呼び出します。

ChangePrinterMode

<設定>

(BL)　= 0: 英語モード
　　　　= 1: 日本語モード
(DX)　= プリンター・ポート（0，1，2）

<結果>

なし

プリンター・モードを（BL）の値に従って切り換えます。プリンターがこのモード変更をサポートしていないときは、何も行わずに戻ります。

第4章　IASによる日本語入力

この章では、日本語入力のためのモジュールとしてInput Assist Subsystem(IAS)が使用されている場合の日本語入力の仕組み、IASが提供するアプリケーション・プログラム・インターフェース(API)、およびそれらのAPIを使用する上での考慮点について説明します。

キーボード入力

OADGでは、以下のキーボードをサポートします。

- IBM 5576-A01キーボード (106キー)
- IBM U.S.English キーボード (101キー)
- AX*仕様キーボード
- 東芝J-3100 *キーボード

キーボードのハードウェアから上がる走査コードは、キーボードによって異なります。この違いを吸収し、アプリケーション、かな漢字変換プログラムに共通のインターフェースを提供するためにいくつかのドライバーがあります。

アプリケーションは、キーボードBIOS(INT 16H)より文字コードを取得することで、キーボードの種類によらないプログラムを書くことができます。

以下にアプリケーションから見たキーボード用のドライバーと、その間のインターフェースを説明します。

DBCS(2バイト文字セット)キーボード・サブシステム

DBCSキーボード・サブシステムは、次のモジュールからなります。

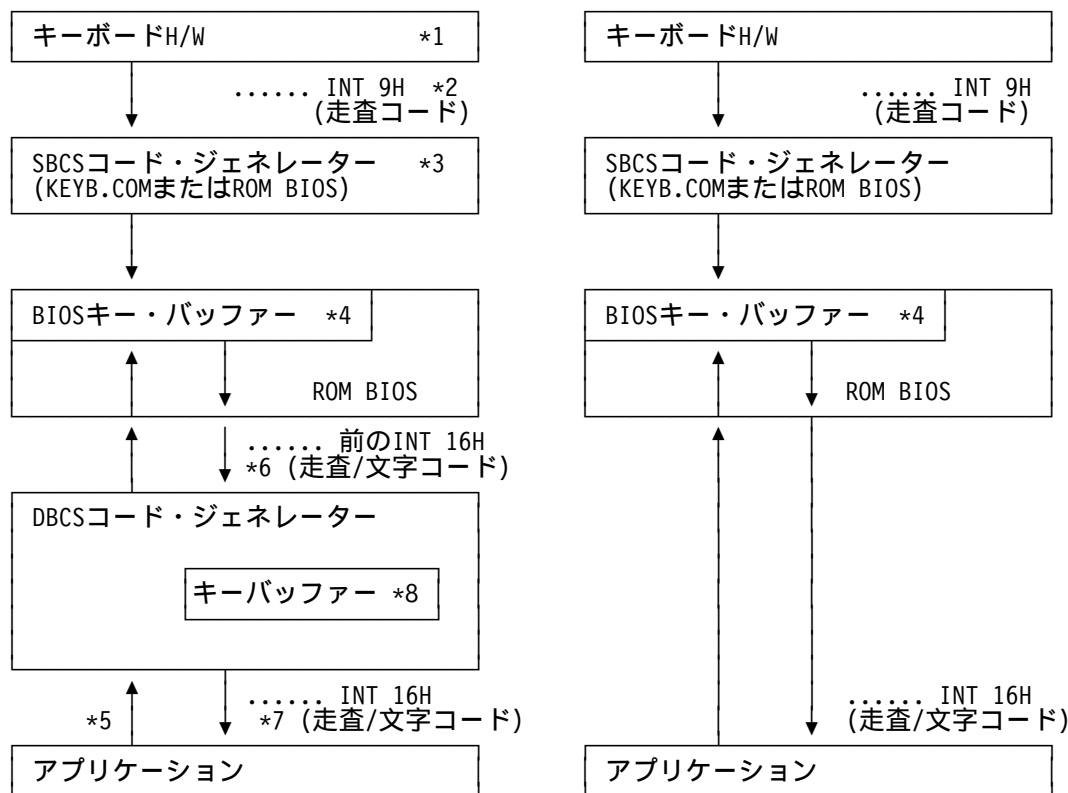
- SBCS(1バイト文字セット)コード・ジェネレーター(ROM BIOS または KEYB.COM)
- DBCS(2バイト文字セット)コード・ジェネレーター

DBCSコード・ジェネレーターは、INT 16Hの割り込みとその処理ルーチンの間に導入されます。(次の図の右は米国などの1バイト・システムの例です。これに対して、DOS/Vでは左図のようにDBCSコード・ジェネレーターが入ります。)

ユーザーが押したキーが、アプリケーションに渡るまでの経路は次のようになります。

キーボード上(*1)のキーが押されたとき、または離されたとき、INT 09Hが発行されます(*2)。INT 09Hで渡されるデータは走査コードです。

INT 09Hは、SBCSコード・ジェネレーターを呼びます。SBCSコード・ジェネレーター(*3)はSBCS関連のシフト状態を管理します。その状態と押されたキーの組み合わせ

により1バイトの文字コード、または機能キーを表す拡張コードを生成します。結果はBIOSキー・バッファーに入れられ(*4)、INT 16Hで読み取られるのを待ちます。

アプリケーションが文字入力のためにINT 16Hを発行する(*5)と、DBCSコード・ジェネレーターが呼ばれます。DBCSコード・ジェネレーターは、かな漢字変換が活動状態でないときには、自分が導入される前のINT 16HのアドレスをFAR CALLする事で、SBCSコード・ジェネレーターの出力結果を取り込み(*6)、アプリケーションにそのまま送ります(*7)。かな漢字変換が活動状態のときは、変換の結果の文字列をキー・バッファー(*8)に蓄えINT 16Hを通じてアプリケーションに1バイトずつ送ります。DBCSコード・ジェネレーターは日本語の入力のシフト状態の管理、表示や、かな漢字変換中の読み、候補の表示も含め、日本語入力のすべてを扱います。

ユーザーがかな漢字変換を始めたとき、アプリケーションの最初のINT 16H (AH=00H/10H)は、かな漢字変換が確定した結果やパススルーのキーを返すか、かな漢字を抜けて別のキーを押すまで、アプリケーションに戻りません。変換結果が確定してはじめて、結果の1バイト目がINT 16に対して値として戻されます。2バイト目以降は、続くINT 16Hに対して1バイトずつ返されます。

日本語入力モードを変更するキー(英数、カナなど)や、かな漢字変換のためのキー(変換、無変換、かな漢字変換中のカーソル・キーなど)は、アプリケーションには渡りません。かな漢字変換の出力結果が、あたかもキーボードから直接入力されたかのように、1バイトずつINT 16Hで取得されます。

キーの解釈方法

IBM 5576-A01キーボードはJIS配列、IBM U.S.English, AX, 東芝J-3100キーボードはASCII配列です。” @ [] \などの特殊記号の配置は両者で異なります。(フランスにはAZERTY配列のようにアルファベットの位置が異なるキーボードもあります。)このように複数の異なるキーボードを一つのアプリケーションでサポートするには、走査コードを読まずに、BIOSが走査コードから生成した文字コードを読むことです。これにより、かな漢字変換プログラムに対する特別な処置も不要となります。

次のように順に検査することで、走査コード(AH)/文字コード(AL)を解釈します。以下の記述でXX、YYは00H以外の8ビット・コードを表します。

1. 00/00ならBreakである
2. 00/XXなら文字コード(XXH)である
3. YY/00またはYY/E0なら拡張コード(YY)である
4. YY/XXなら文字コード(XXH)である

文字コードは、制御コード、1バイト文字コード、2バイト文字コードの1バイト目または2バイト目のいずれかです。

1. 前に読んだコードが2バイト文字の1バイト目なら、次のコードは2バイト目
2. 00H-1FH, 7FHなら制御コード
3. DBCSベクターの範囲内(DOS/Vでは、81H-9FH、E0H-FCH)なら、2バイト文字の1バイト目
4. それ以外なら1バイト文字(英数カナ)

1バイトの英数カナ文字は、文字コードと走査コードを取得できますが、走査コードを使うと、それぞれのキーボードに対応する処置が必要になります。

日本語入力とかな漢字変換

DBCSコード・ジェネレーター

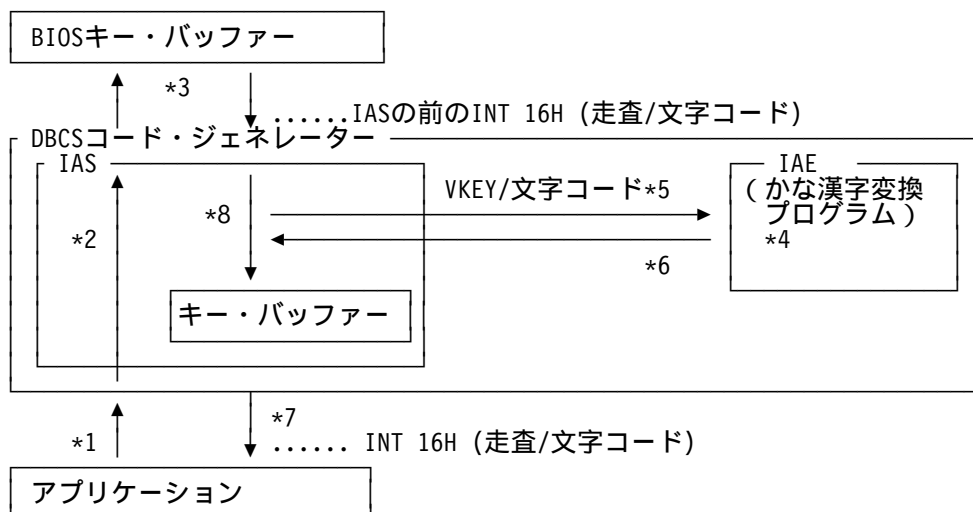
この節は、ハードウェアに依存しないかな漢字変換プログラムのプログラミング考慮点を説明します。

OADG標準のDBCSコード・ジェネレーターは次の2つからなります。

- 入力支援サブシステム(IAS)
- 日本語入力エディター(IAE、かな漢字変換プログラム)

IASがキーボード、アプリケーションとのインターフェースと、日本語入力モードの管理をおこないます。IAEはかな漢字変換をおこないます。

IASを利用することで、キーボードに依存しないかな漢字変換プログラムを作ることができます。IASとIAEの両方を置き換える形でかな漢字変換フロントエンド・プロセッサー(FEP)を作る場合は、FEPがサポートするキーボードの違いを吸収しなければなりません。

アプリケーションがINT 16Hで文字を読み取る場合、この両者に違いはありません。

アプリケーションが文字入力のためにINT 16Hを発行する(*1)と、IAS(*2)が呼ばれます。IASは日本語入力モードの管理と、かなコードの生成を行います。SBCSコード・ジェネレーターの出力結果を、IASが導入される前のINT 16HのアドレスをFAR CALLすることで取り込みます(*3)。現在のモードと、押されたキーの組み合わせで、カタカナ、ひらがなの文字コードを生成します。かな漢字変換を行うときは、このかなコードがIAE(*4)に送られます(*5)。かな漢字変換を行うために必要な機能キーは、

IASが疑似走査コードからVKEYと呼ばれる仮想キーに変換してIAEに送ります。これによりキーボードに依存しないかな漢字変換プログラムを作ることができます。IAEは変換された漢字混じりの文字列を返します(*6)。IAEはこれを蓄え、INT 16Hを通じてアプリケーションに1バイトずつ送ります(*7)。IAEが活動状態でないときは、SBCSコード・ジェネレーターからの1バイト文字、機能キー、IASで生成されたかな文字キーがバッファーに蓄えられ(*8)、1バイトずつアプリケーションに返されます。DBCSシフト状態の表示や、かな漢字変換中の読み、候補の表示もIASが行います。

IAEは、かな漢字変換を行います。IASからは文字コードと、機能キーを表す仮想キーが送られます。一連の文字キーと機能キーでかな漢字変換を行い、結果を文字列としてIASに返します。モードの管理はIASが行いますが、IAEもモード変更の指示をすることができます。モード変更キーが押されたとき、IASはまず仮想キーをIAEに送ります。IAEが、このキーに対してモード変更をIASに指示します。IAEが特に指示しなかった場合は、IASがモードを変更します。

入力支援サブシステム (IAS)

日本語入力サブシステム(以後IAS)は日本語入力を扱います。INT 16Hの割り込みを取り込むことで、アプリケーションとSBCSコード・ジェネレーターの間に位置します。IASは次の機能を持っています。

- 日本語入力モードの管理
  英数/かな/カナ、半角/全角、ローマ字入力、漢字(FEPのオン、オフ)のような日本語入力に関するモードの管理はIASで行います。画面最下行に日本語入力キーボード・シフト状況を表示します。またアプリケーションからのモード制御用のINT 16H AH=13H, 14Hを処理します。

- かなコードの生成
  半角カタカナ、全角カタカナ、全角ひらがな、かな記号、全角英数などの文字コードは、IASが生成します。

- 日本語入力エディターのインターフェース処理
  日本語入力エディター(IAE)が活動状態のときは、IASの生成したかな文字と、仮想キー(VKEY)に変換された機能キーをIAEに渡します。変換中の表示データをIAEから受け取り画面に表示します。モード変更キーはIAEに伝え、IAEがモードを決定する権利を持ちます。IAEの指示がない場合はIASがモード変更を行います。

- アプリケーションからのINT 16Hの処理
  IAEからの出力をバッファーに保持し、INT 16Hを通じてアプリケーションに1バイトずつ渡します。変換などの日本語入力関係のキーは通常アプリケーションに渡たりませんが、これらのキーを渡すようアプリケーションから設定することもできます。詳細は“入力支援サブシステムバージョン2.0技術解説書”を参照してください。

日本語入力エディター(IAE)

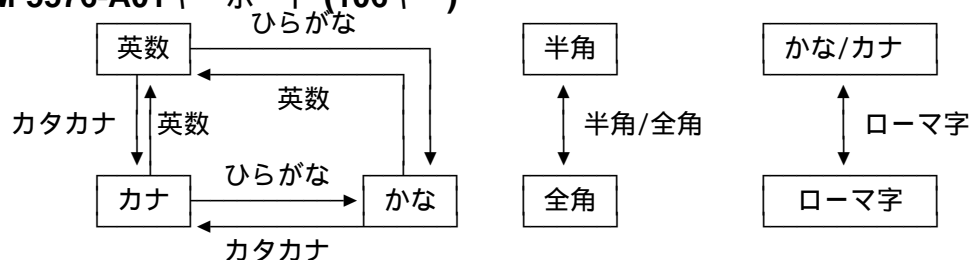
いわゆるかな漢字変換プログラムです。IASがキーボード、アプリケーション、ディスプレイ、とのインターフェースを行うので、IAEはこれらに依存しないプログラムとして作成することができます。

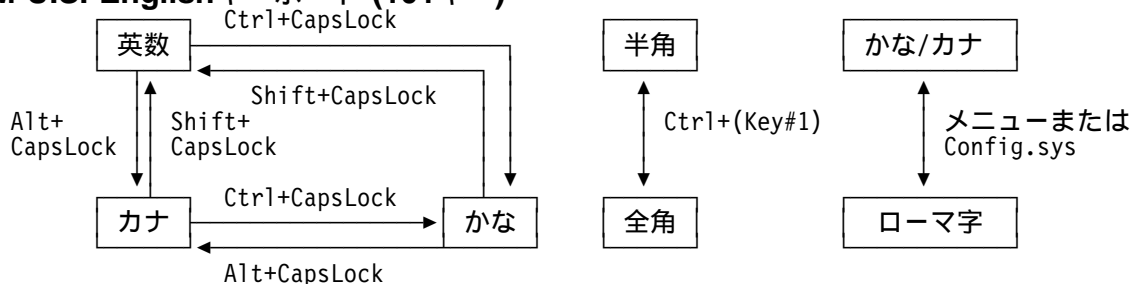
かなの文字コードと、機能キーを表す仮想キーをIASから受け取ります。読み、候補などの変換中の文字表示はIASに送って、IAS側で表示します。日本語入力のモードはIASが管理しますが、モード変更キーは一度IAEに渡ってきますので、IAEで指示することができます。IAEの指示によって、IASがモードを変更し、画面の最下行に表示します。変換結果はIASに返し、IASがアプリケーションとのインターフェースを行います。

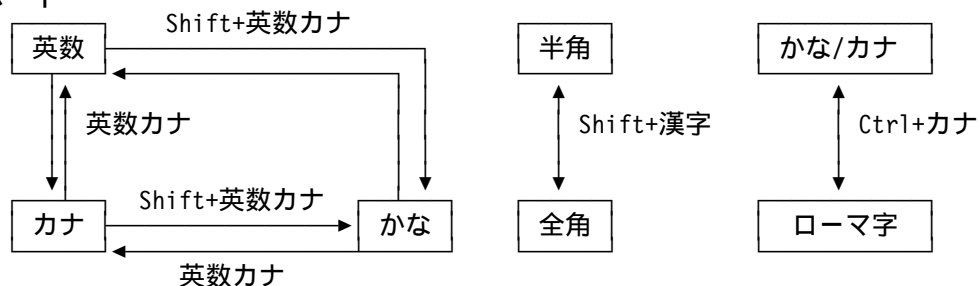
IAEとIASのインターフェース詳細は“入力支援サブシステムバージョン2.0技術解説書”を参照してください。

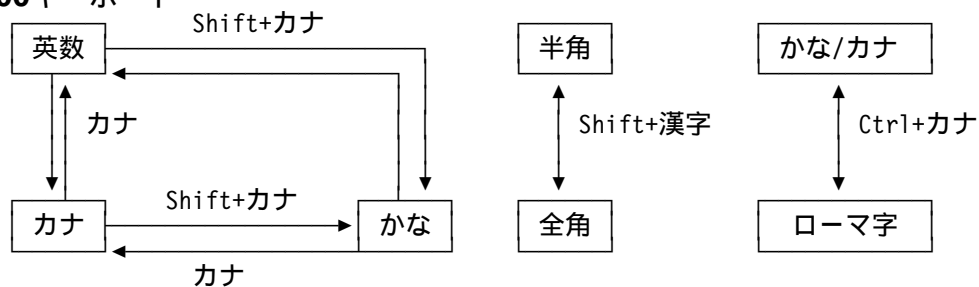
日本語入力モード

4種類のキーボードにおける日本語入力モードの遷移は次のとおりです。日本語入力モードの管理はIASが行うので、それぞれのキーボードをサポートしたIASを導入する必要があります。アプリケーションからはこのモードと遷移を気にする必要はありません。

IBM 5576-A01キーボード (106キー)

- 3つのブロックは独立です

IBM U.S. Englishキーボード (101キー)

- 3つのブロックは独立です

AXキーボード

- 3つのブロックは独立です
- かなモード、半角/全角、ローマ字モードの切り替えはオプションです。

東芝J-3100キーボード

- 3つのブロックは独立です
- カナ・キーの刻印が異なる以外はAXキーボードと同じです。
- かなモード、半角/全角、ローマ字モードの切り替えはオプションです。

日本語入力モードに関するキーを各キーボードごとに示します。これらのキーはIASが入力モードを制御するのに必要なキーです。

下表の*1の機能、及びその他の日本語入力モード用キー(変換、無変換など)のキー位置、機能はDBCSコード・ジェネレーター(IAEまたはFEP)に依存します。

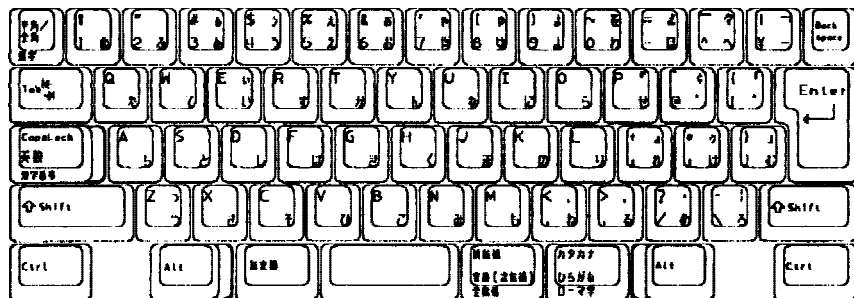
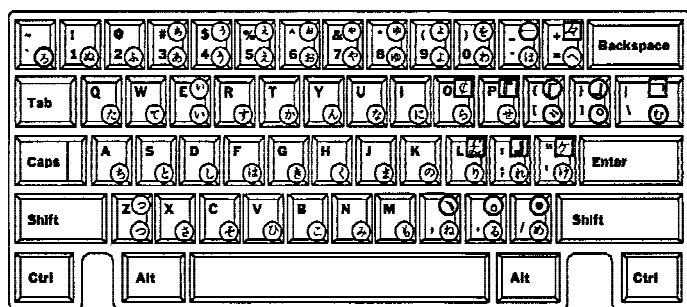
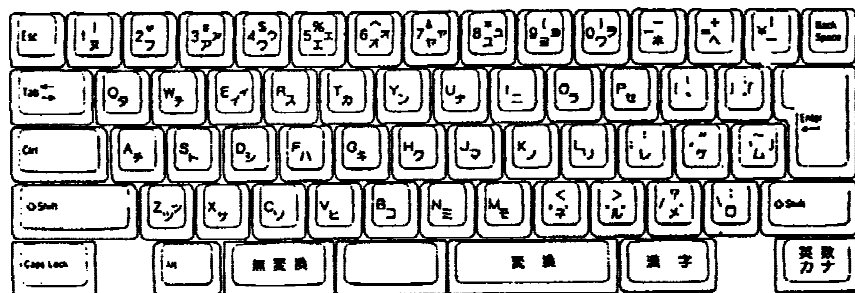
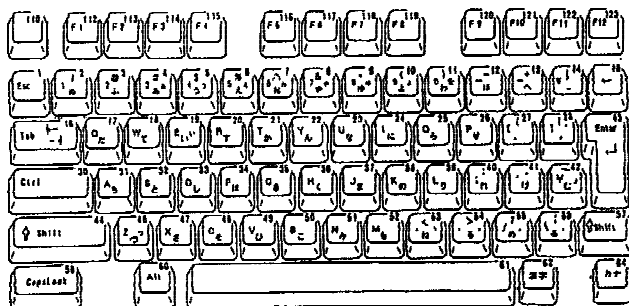
各機能に対するキーをシフト状態+キー番号で示しました。機能名がキートップに書かれている場合は、その機能名を横に示します。機能名が書かれていない場合、そのキーのキートップをかっこ内に示します。

機能	IBM 5576-A01ｷｰﾎﾞｰﾄﾞ	IBM U.S. Eng.ｷｰﾎﾞｰﾄﾞ	AXキーボード	J-3100キーボード
漢字	Alt+ #1 漢字	Alt+ #1 (`)	#62 漢字	#62 漢字
英数	#30 英数	Shift+#30 (CapsLock)		
カタカナ	Shift+#133 カタカナ	Alt+ #30 (CapsLock)		
ひらがな	#133 ひらがな	Ctrl+ #30 (CapsLock)		
英数⇔カタカナ ひらがな⇒カタカナ			#64 英数カナ	#64 カナ
英数⇔ひらがな *1 カタカナ⇒ひらがな			Shift+#64 (英数カナ)	Shift+#64 (カナ)
ローマ字オン⇔オフ *1	Alt+ #133 ローマ字	メニュー/Config *2	Ctrl+ #64 (英数カナ)	Ctrl+ #64 (カナ)
半角⇔全角 *1	#1 半角/全角	Ctrl+ #1 (`)	Shift+#62 (漢字)	Shift+#62 (漢字)
かな漢字制御 *1	Ctrl+ #1	Alt+Ctrl+#1 (`)		

注: *1の機能はIAEまたはFEPに依存します。
*2は、かな漢字制御キーでメニューを表示して変更するか、CONFIG.SYSにIASのデフォルト値を書いて指定します。

かなキー配列

次の図はIASが生成するかな文字の配置を示します。

IBM 5576-A01キーボード

IBM U.S. Englishキーボード

AXキーボード

東芝J-3100キーボード

日本語入力API

INT 16H キーボード入出力

AH=13H　２バイト文字セットの状況モードの設定と読み取り

2バイト文字セットの状況モードを設定（変更）したり、読み取ることができます。

＜設定＞

(AL)　設定または読み取り

= 00H　2バイト文字セットのキーボード状況モードを設定します。

= 01H　2バイト文字セットのキーボード状況モードを読み取ります。

(DH)　予約済み（変更しないでください）

(DL)　状況モード

ビット7 = 0	非漢字モード
= 1	漢字モード
ビット6 = 0	ローマ字オフ
= 1	ローマ字オン
ビット5	予約済み
ビット4，3 = 00	シフト状況を25行目に表示する
= 01	予約済み
= 10	シフト状況を25行目にトグル方式で表示する
= 11	シフト状況を26行目に表示する
ビット2，1 = 00	英数シフト
= 01	カタカナ・シフト
= 10	ひらがなシフト
= 11	予約済み
ビット0 = 0	半角
= 1	全角

注:

1. この機能は、入力支援サブシステム（$IAS.SYS）により提供されます。入力支援サブシステムが導入されていない場合は、BIOSは省略時のシフト状況（DX=0）を返します。ただし、かな漢字変換プログラム（たとえば、$IAESKK.SYS）が導入されていないと、ビット7は無効（常に0）です。

2. ビット3，4で設定する機能はすべて、現在の画面モードが03Hまたは73Hのときにのみ有効です。他のモードのとき、この機能は無視されます。実際に切り換わったかどうかは、AL=00Hで設定後、もう一度AL=01Hで読み取り、確認してください。

   シフト状況を25行目に表示する（またはトグル方式の）モードと、26行目に表示するモードの間で切り換えを行うと、ビデオ・モード設定を内部的に呼び出します。

3. トグル方式による表示（トグル・モード）では、シフト状況の表示は、以下のようになります。

シフト状況表示の初期状態: INT 16H，AX=1300Hによりトグル・モードに設定したときの初期状態は以下のとおりです。

- 英数・半角・非漢字状態（DL=10H/50H）に設定した場合、シフト状況は表示されません。
- 上記以外に設定した場合、シフト状況は画面最下行（25行）に表示されます。

シフト状況の表示: シフト状況表示は、次のいずれかの場合に更新されます。

- INT 16H，AX=1300Hが発行されたとき。ただし、設定の前後でシフト状況に変化がない場合は、更新されません。
- キーボードからシフト状況を制御するキー入力があった場合。シフト状況とは以下の入力状態を意味します。

  数字/カタカナ/ひらがな状態
  半角/全角状態
  ローマ字入力状態
  漢字/非漢字モード入力状態

シフト状況の消去: 表示されたシフト状況は、次の場合に消去されます。

- INT 16H，AH=00H/10Hを発行したアプリケーションに、入力コードが返される直前。

トグル・モードは、２バイト文字を入力するまでシフト状況を表示する必要のない場合などに使用されます。

トグル・モードは、標準テキスト・モード（80×25のモード 03H，73H）の場合にのみ有効です。

第5章　DOSV.INIファイル

DOS/V初期設定ファイル(DOSV.INI)には、DOS/V環境の初期化のための、DOS/V関連モジュールのパラメーターに関する情報が含まれています。たとえば、DOSV.INIファイルにはDOS/Vフォント・サブシステムがロードするフォント・ファイル名が含まれます。

DOSV.INIファイルは、DOSの導入プログラムによって作成されます。$DISP.SYSなどのDOS/V関連モジュールは、初期化の時点でDOSV.INIファイルを参照します。ユーザーがDOS/V環境をカスタマイズする必要がある場合は、通常、DOS/Vカスタマイズ・プログラム(SETUPV.EXE)を始動して、DOSV.INIファイルを更新します。

DOS/V初期設定ファイルのフォーマット

DOSV.INIファイルは、編集可能なテキスト・ファイルです。DOSV.INIファイルにはいくつかのブロックがあり、各ブロックはセクションといくつかのキー名から構成されます。ひとつのブロックのフォーマットは、次のとおりです。

```
; コメント
[セクション]
(サブセクション)
キー名1=....
キー名2=....
   :
   :
```

まずセクション名を '[' と ']' で囲んで定義します。'[' は行の最初のカラムになければなりません。サブセクションを定義する場合は '(' と ')' で囲みます。この場合も、 '(' は行の最初のカラムになければなりません。DOSV.INIファイルにはコメントを入れることができます。最初のカラムがセミコロン(;)になっていると、その行がコメント行であることを示します。DOSV.INIファイルの中では、英字に関して、大文字と小文字の両方を使用することができますが、DOSV.INIファイルを参照するプログラムは大文字と小文字を区別しません。

各セクションとキー名を以下に示します。

[Font]セクション

```
[Font]

FontPath=パス名

SBCSxxyy_n=Enabled/Disabled/ROM,フォント・ファイル名,[フォント名]
DBCSxxyy_n=Enabled/Disabled/ROM,フォント・ファイル名,[フォント名]

UDCxxyy=Enabled/Disabled,フォント・ファイル名

SPCLxxyy=Enabled/Disabled,フォント・ファイル名

UDCRange=xxxx,yyyy

Method=AUTO/INT15/XMS

SKKDic=SKK用辞書名
```

[Font]セクションは、DOS/Vフォント・サブシステム($FONT.SYS)が参照します。
[Font]セクションの各キー名は、次のとおりです。

FontPath

SBCSxxyy_n、**DBCSxxyy_n**、および**UCDxxxx**が指定する全フォント・ファイルが含まれるディレクトリーのパス名です。パス名は必ず '¥' 記号で終るように指定してください。デフォルト（すなわち、何も指定がなかった場合）は、次のパス名が使用されます。

```
FontPath=C:¥
```

SBCSxxyy_nまたはDBCSxxyy_n

これは、半角フォント・パターンおよび全角フォント・パターンを含むフォント・ファイルの名前です。ここで、'xx' と 'yy' はそれぞれ文字フォントの幅と高さです。また、'n'は1を基準とするフォント・インデックスです。フォント・インデックスとは、そのフォント・ファイルが SBCSxxyy（またはDBCSxxyy) のフォントの何番目のフォントであるかを示したものです。'_n'を省略した場合、省略時フォントを指定したことになります。ただし、フォント・インデックスは、フォント・サブシステムとして**$FONT.SYS バージョン 1.30** 以降が使用されている場合のみ機能します。

フォント名は、フォントの名称またはタイプを表します。

このパラメーターの最初の値には、Enabled、Disabled、またはROMを指定できます。２番目の値(フォント・ファイル名)は、最初の値をEnabledまたはROMと指定した場合にのみ参照されます。

Enabled　フォント・データがメモリーにロードされることを示します。

Disabled　フォント・データがメモリーにロードされないことを示します。

Rom フォント・データがコンピューターに組み込まれたフォントROMの中に含まれる場合は、そのフォント・データを優先して使用することを示します。ROMの中のデータを無視させたい場合は、Enabledを指定してください。

デフォルトは、少なくとも次の値が仮定されます。

```
SBCS0816=ROM,$JPNHN16.FNT
SBCS0819=ROM,$JPNHN19.FNT
DBCS1616=ROM,$JPNZN16.FNT
```

12ドットのフォント・パターンが必要な場合は、次のように、さらに12ドットのフォント・ファイルを指定する必要があります。

```
SBCS0612=Enabled,$JPNHN12.FNT
DBCS1212=Enabled,$JPNZN12.FNT
```

24ドットのフォント・パターンが必要な場合は、次のように、さらに24ドットのフォント・ファイルを指定する必要があります。

```
SBCS1224=Enabled,$JPNHN24.FNT
DBCS2424=Enabled,$JPNZN24.FNT
```

あるフォント・パターンを特定のフォント・インデックスに登録する場合、次のように指定します。(この例では、24ドットの楷書フォントをフォント・インデックス3に指定します。)

```
SBCS1224_3=Enabled,JKAIH24N.FNT, 楷書
DBCS1224_3=Enabled,JKAIZ24N.FNT, 楷書
```

UDCxxyy

これは、幅が'xx'で高さが'yy'のユーザー定義文字フォント(UDC: User-Defined Character)の名前です。デフォルトは、少なくとも次の値が仮定されます。

```
UDC1616=Enabled,$SYS1Z16.FNT
UDC2424=Enabled,$SYS1Z24.FNT
```

SPCLxxyy

全角特殊フォントをロードするためのものです。全角特殊フォントとは、従来のDOS/Vでは定義されていないが他のDOSなどで定義されている文字のフォントです。SPCL1212=で12x12ドットの全角特殊フォント、SPCL1616=で16x16ドットの全角特殊フォント、SPCL2424=で24x24ドットの全角特殊フォントをそれぞれ指定します。例えば、IBM DOS/V Extension V2.0 では、NECがJIS区点コードで13区に定義しているフォントと同等のもの(83文字)を以下のファイルで提供しています。

```
JSPCL12N.FNT (12x12ドット・フォント)
JSPCL16N.FNT (16x16ドット・フォント)
JSPCL24N.FNT (24x24ドット・フォント)
```

これらのフォントをDOSV.INIファイルに指定するには、以下のように記述します。

```
SPCL1212=Enabled,JSPCL12N.FNT
SPCL1616=Enabled,JSPCL16N.FNT
SPCL2424=Enabled,JSPCL24N.FNT
```

この全角特殊フォントは、あくまでも他のDOS上で定義された文字のフォントを一時的に表示・印刷したりするためのもので、ホスト環境や OS/2 のウィンドウ表示では使用できないなどの制限があります。また、全角特殊フォントはユーザー・フォントと同様、ロードできるフォントは1つのサイズにつき1種類だけです。つまり、明朝、ゴシック、楷書といった複数の種類のフォントを同時にロードすることはできません。

全角特殊フォントの指定は、フォント・サブシステムとして**$FONT.SYSバージョン 1.30**以降が使用されている場合のみ機能します。

UDCRange

このパラメーターは、ユーザー定義文字フォントの開始コードを16進数で、文字数を10進数で指定します。指定されたユーザー定義文字の数は、94の倍数に切り上げられます。デフォルトは次のようになります。

```
UDCRange=F040,658
```

最初のパラメーター（開始コード）は'F040'に固定されており、変更できません。２番目のパラメーター（ユーザー定義文字の数）のみをユーザーは指定できます。２番目のパラメーターは、CONFIG.SYSの中の$FONT.SYSの/U=パラメーターと同じです。

Method

このパラメーターは、$FONT.SYSがどのようにフォント・パターンのために拡張メモリーを割り振るか、また拡張メモリーと基本メモリーとの間でどのようにフォント・パターンを移動するかを指定します。有効な値は次のとおりです。

AUTO　メモリー割り振りの方法は$FONT.SYSが自動判定します。

INT15　$FONT.SYSは、INT 15Hの方法を使用します。

XMS　$FONT.SYSはXMS方法を使用します。

デフォルトの設定は次のとおりです。

SKKDic

単漢字変換（$IAESKK.SYS）が使用する辞書ファイルを指定します。このファイルは、キー名 **FontPath** で指定される**ドライブ:パス** になければなりません。デフォルトは次のファイルが仮定されます。

```
FNT
```

[Display]セクション

```
[Display]

HDWRScroll=AUTO/ON/LC/OFF

TextBufferSize=xxxx
```

[Display]セクションは、DOS/Vビデオ・サブシステム($DISP.SYS)が参照します。[Display]セクションの各キー名は、次のとおりです。

HDWRScroll

このパラメーターは、エミュレート・テキスト・モード、03H、73Hにおいて、$DISP.SYSがどのように画面をスクロールするかを指定します。有効な値は次のとおりです。

AUTO スクロールの方法は$DISP.SYSが自動判定します。

ON $DISP.SYSは、APA開始アドレス・レジスターのみを使用します。

LC $DISP.SYSは、APA開始アドレス・レジスターとライン・コンペア・レジスターを使用します。

OFF $DISP.SYSは、APA開始アドレス・レジスター、ライン・コンペア・レジスターのいずも使用しません。画面のスクロールは、単に画面イメージ・データのコピーによってのみ行います。

デフォルトの設定は次のとおりです。

```
HDWRScroll=AUTO
```

TextBufferSize

このパラメーターは、テキスト・バッファー・サイズの最大値を指定します。最大値は10進数で指定します。テキスト・バッファーとは、$DISP.SYSがテキスト・モードにおいて、文字属性を格納するメモリー領域のことです。このパラメーターは、ユーザーがIBM DOS/V Extensionを導入して、高密度テキスト・モードを使用する場合に必要です。

このパラメーターは、CONFIG.SYSの中の$DISP.SYSの/TS= パラメーターと同じです。

[Keyboard]セクション

```
[Keyboard]

Type=JP/US/AX/J3
```

[Keyboard]セクションは、DOS/V入力支援サブシステム($IAS.SYS)が参照します。[Keyboard]セクションのキー名は、次のとおりです。

Type

このパラメーターは、接続されているキーボードの型を指定します。有効な値は次のとおりです。

JP 日本語106キーボート(5576-001/002/003/A01/B01/C01)

US 米国英語101キーボード

AX AXキーボード

J3 東芝 J3100ノートブック型キーボード

デフォルトの設定は次のとおりです。

このパラメーターは、CONFIG.SYSの中の$IAS.SYSの/K= パラメーターと同じです。

[Input]セクション

```
[Input]

Load=AUTO/BASE

Roman=ON/OFF

GrphOnTheSpot=ON/OFF

DisplaySSL=26/25/Toggle
```

[Input]セクションは、DOS/V入力支援サブシステム($IAS.SYS)が参照します。[Input]セクションの各キー名は、次のとおりです。

Load

このパラメーターは、どこに$IAS.SYSをロードするかを指定します。有効な値は次のとおりです。

AUTO EMSメモリーが使用可能な場合、IASはEMSメモリーにロードされます。十分なEMSメモリーが無い場合、IASは基本メモリーにロードされます。

BASE EMSメモリーが使用可能であっても、IASは基本メモリーにロードされます。

デフォルトの設定は次のとおりです。

このパラメーターは、CONFIG.SYSの中の$IAS.SYSの/X= パラメーターと同じです。

Roman

このパラメーターは、IASの初期状態を、ローマ字入力モードにするかどうかを指定します。有効な値は次のとおりです。

ON 初期状態がローマ字入力モードになります。

OFF 初期状態がローマ字入力モードになりません。

デフォルトの(Roman=を指定しなかった場合の)設定は次のとおりです。

このパラメーターは、CONFIG.SYSの中の$IAS.SYSの/R= パラメーターと同じです。

GrphOnTheSpot

このパラメーターは、グラフィック・モードで、スポット変換を有効にするかどうかを指定します。有効な値は次のとおりです。

ON グラフィック・モードで、スポット変換を有効にします。

OFF グラフィック・モードで、スポット変換を無効にします。

デフォルトの設定は次のとおりです。

```
GrphOnTheSpot=OFF
```

このパラメーターは、CONFIG.SYSの中の$IAS.SYSの/G= パラメーターと同じです。

DisplaySSL

このパラメーターは、シフト状況表示域(SSL: Shift Status Line)を「どこに」「どのように」表示するかを指定します。有効な値は次のとおりです。

26 IASは、可能であれば画面モードを26行モードに変更し、最下位行をシフト状況表示域に予約します。最下位行以外の残りの25行は、アプリケーションが使用できます。

25 IASは、25行画面の25行目にシフト状況を表示します。アプリケーションは25行目を使用しないか、状況表示とオーバーラップさせて表示します。このモードは従来のバージョンと互換のモードです。

Toggle システムが「かな漢字入力モード」になるまで、シフト状況表示域は予約されません。「かな漢字入力モード」を抜けると、最下位行が解放されて、その部分の元の表示データが復元されます。ただし、「かな漢字入力モード」でなくても、日本語シフト・キー（［ひらがな］キー、［半角/全角］キーなど）を押すと、モードの変更をユーザーに知らせるために、変更後のモードを最下位行に表示します。表示は、次の日本語シフト・キー以外のキー入力で消去されます。

デフォルトの設定は次のとおりです。

```
DisplaySSL=25
```

[Printer]セクション

```
[Printer]

(LPT1)

PrinterDriver=PRNIBM.COM/PRNESCP.COM/EPRNIBM.COM/EPRNESCP.COM/
EPCLIBM.COM/EPCLESCP.COM/NONE

UDCBuffer=xxxx[,yyyy]

UDCImage=Loaded/NotLoaded

DriverStatus=Enabled/Disabled

Exchange=ON/OFF

IBMJType=24/32/48

EPrnType=PRO/LQ

PaperWidth=Narrow/Wide

PaperSize=LTR/A4

(LPT2)
  :

(LPT3)
  :
```

[Printer]セクションは、DOS/Vプリンター・ドライバー・マネージャー($PRN.SYS)と個々のプリンター・ドライバーが参照します。[Printer]セクションの各キー名は、次のとおりです。

LPT1, LPT2, LPT3

LPT1/LPT2/LPT3は、サブセクションです。LPTxキー名の x は、プリンターが接続されるポート番号を指定します。

PrinterDriver

PrinterDriverは、各ポート毎に組み込まれるプリンター・ドライバーを指定します。有効な値は以下のとおりです。

PRNIBM.COM
PRNESCP.COM
EPRNIBM.COM

EPRNESCP.COM
EPCLIBM.COM
EPCLESCP.COM
NONE

UDCBuffer

UDCBufferは、ユーザー定義文字を印刷するためのバッファーを、文字コードまたは文字コードの範囲で指定します。このパラメーターは、キー名PrinterDriverがPRNESCP.COMの場合のみ有効です。

UDCImage

UDCImageは、ユーザー定義文字のフォント・イメージを文字印刷のたびにプリンターに送るか（NotLoaded）、あらかじめすべてのユーザー定義文字のフォント・イメージをプリンターに送っておくか（Loaded）を指定します。

Loadedを指定するには、あらかじめUSRFNTコマンドでユーザー定義文字のフォント・イメージをプリンターに登録しておく必要があります。

デフォルトの値は、NotLoadedです。このパラメーターは、キー名PrinterDriverがPRNIBM.COMで、かつIBMJTYPEが32または48の場合にのみ有効です。

DriverStatus

DriverStatusは、各ポート毎に組み込まれるドライバーを、有効にするか(Enabled)無効にするか(Disabled)を指定します。無効にすると、ドライバーはINT 17Hで印刷するデータに対し、何ら処理を行いません。

Exchange

Exchangeは、JIS X0208-1983でコード・ポインターの変更が行われた26組の漢字に対して、シフトJISコードからJIS 16進コードへの変換を制御します。有効な値は以下のとおりです。

ON　シフトJISコードをJIS 16進コードに変換します（26組の入れ替えを行います）。

OFF　シフトJIS83並びのシフトJISコードをJIS 16進コードに変換します（26組の入れ替えは行いません）。

デフォルトの値は、OFFです。このパラメーターは、キー名PrinterDriverがPRNESCP.COMの場合にのみ有効です。

IBMJType

IBMJTypeは、IBM PS/55プリンターの種類を指定します。有効な値は以下のとおりです。

24　IBM 5575/5577系 24ドット・プリンター

32　IBM 5587系 32ドット・プリンター

48　IBM 5584系 48ドット・プリンター

このパラメーターは、キー名PrinterDriverがPRNIBM.COMの場合にのみ有効です。

EPrnType

EprnTypeは、英語プリンターの実タイプ（エミュレートする論理的なプリンター・タイプではなく、実際のプリンター・タイプ）を指定します。有効な値は以下のとおりです。

PRO IBM Personal Printer Series II(24ドット) またはProprinter X24E/XL24E

LQ EPSON LQ 1050 （ESC/P 24ドット英語プリンター）

このパラメーターは、キー名PrinterDriverがEPRNIBM.COMまたはEPRNESCP.COMの場合にのみ有効です。

PaperWidth

PaperWidthは、プリンターが処理できる用紙の幅を指定します。Narrowは、最大幅が8インチ以下用のプリンターに対して指定します。Wideは、最大幅が13.6インチ以下用のプリンターに対して指定します。このパラメーターは、キー名PrinterDriverがEPRNIBM.COMまたはEPRNESCP.COMの場合にのみ有効です。

PaperSize

PaperSizeは、プリンター用紙のサイズを指定します。指定できる値は、A4またはLTR（レター・サイズ）です。このパラメーターは、キー名PrinterDriverがEPCLIBM.COMおよびEPCLESCP.COMのときのみ有効です。

DOSV.INIファイルの検索優先順位

DOSV.INIファイルをアクセスするDOSのモジュールは、ファイルの位置を知る必要があります。ファイル検索の優先順位は次のとおりです。

デバイス・ドライバーの場合

DOSV.INIファイルをアクセスするデバイス・ドライバーは次の順番でファイルを検索します。

1. CONFIG.SYSの /INI= パラメーターで指定されたディレクトリー
2. デバイス・ドライバーが存在するディレクトリー

1. CONFIG.SYSの /INI= パラメーターで指定されたディレクトリー

たとえば、CONFIG.SYSの中の次のステートメントは、DOSV.INIファイルが'C:¥DOSV'ディレクトリーに存在することを示しています。

```
DEVICE=C:¥DOS¥$DISP.SYS   /INI=C:¥DOSV¥
```

2. デバイス・ドライバーが存在するディレクトリー

たとえば、CONFIG.SYSの中に次のように指定された場合は、DOSV.INIファイルが'C:¥DOS'ディレクトリーになければなりません。

```
DEVICE=C:¥DOS¥$DISP.SYS
```

実行プログラムの場合

DOSV.INIファイルをアクセスする実行プログラムは次の順番でファイルを検索します。

1. /INI= コマンド・パラメーターで指定されたディレクトリー
2. 実行プログラムが存在するディレクトリー

1. /INI= コマンド・パラメーターで指定されたディレクトリー

たとえば、次のコマンドは、DOSV.INIファイルが'C:¥DOSV'ディレクトリーに存在することを示しています。

```
C:¥DOS¥SETUPV.EXE   /INI=C:¥DOSV¥
```

2. 実行プログラムが存在するディレクトリー

たとえば、次のコマンドが実行される場合は、DOSV.INIファイルが'C:¥DOS'ディレクトリーになければなりません。

```
C:¥DOS¥SETUPV.EXE
```

その他の考慮事項

DOSV.INIファイルの中のいくつかのパラメーターは、たとえば$DISP.SYSに対する/TS= パラメーターのように、すでにCONFIG.SYSの中のステートメントのパラメーターとして定義されているものもあります。このようなパラメーターは、ユーザー環境の中で、DOSV.INIファイルとCONFIG.SYSファイルの間で矛盾する可能性もあります。この場合、CONFIG.SYSに指定されたパラメーターが、DOSV.INIに指定されたものより優先され、DOSV.INIのパラメーターは無視されます。

索引

日本語、英字、数字、特殊文字の順に配列されています。なお、濁音と半濁音は清音と同等に扱われています。

〔ア行〕

〔カ行〕

〔サ行〕

〔タ行〕

〔ナ行〕

〔ハ行〕

〔マ行〕

〔ワ行〕

A

C

D

E

F

I

L

M

O

P

R

S

T

V

W